



# Fundamentals of Web Technology

Anil Kumar Yadav and Vinod Kumar Yadav



# **Fundamentals of Web Technology**



# **Fundamentals of Web Technology**

**Anil Kumar Yadav and Vinod Kumar Yadav**



[www.arclerpress.com](http://www.arclerpress.com)

# **Fundamentals of Web Technology**

*Anil Kumar Yadav and Vinod Kumar Yadav*

## **Arcler Press**

224 Shoreacres Road

Burlington, ON L7L 2H2

Canada

[www.arclerpress.com](http://www.arclerpress.com)

Email: [orders@arclereducation.com](mailto:orders@arclereducation.com)

## **e-book Edition 2022**

ISBN: 978-1-77469-367-4 (e-book)

This book contains information obtained from highly regarded resources. Reprinted material sources are indicated and copyright remains with the original owners. Copyright for images and other graphics remains with the original owners as indicated. A Wide variety of references are listed. Reasonable efforts have been made to publish reliable data. Authors or Editors or Publishers are not responsible for the accuracy of the information in the published chapters or consequences of their use. The publisher assumes no responsibility for any damage or grievance to the persons or property arising out of the use of any materials, instructions, methods or thoughts in the book. The authors or editors and the publisher have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission has not been obtained. If any copyright holder has not been acknowledged, please write to us so we may rectify.

**Notice:** Registered trademark of products or corporate names are used only for explanation and identification without intent of infringement.

© 2022 Arcler Press

ISBN: 978-1-77469-189-2 (Hardcover)

Arcler Press publishes wide variety of books and eBooks. For more information about Arcler Press and its products, visit our website at [www.arclerpress.com](http://www.arclerpress.com)

## ABOUT THE AUTHORS



**Dr. Anil Kumar Yadav** received his PhD in Computer Science and Engineering and his M.Tech in Information Technology, B.Tech in Computer Science and Engineering. Presently, he is working as an Associate Professor in Computer Science & Engineering Department at IES College of Technology Bhopal M.P. His major areas of interest include Application-Oriented Reinforcement Learning, Game Technology and Machine Learning. He received many academic related awards. He has published 03 Patent, one text book on “Data Structures with C program,” ARCLER PRESS and more than 34 research publication at National and International Level.



**Mr. Vinod Kumar Yadav** Is Pursuing Phd (Cse) From Rntu, Bhopal And Completed His M.tech In Computer Science And Engineering From The University Of Rajiv Gandhi Proudyogiki Vishwavidyalaya Bhopal M.p. And B.tech In Information Technology From Guru Ghasidas Vishwavidyalaya Central University Bilaspur C.g. (India). He Is Also Gate Qualified In Information Technology. At Present He Is Working As Head Of Department ,Computer Science & Engineering Bmct, Bhopal (M. P.). His Areas Of Interests Are In Data Structures, Data Mining, Artificial Intelligence, Machine Learning, Deep Learning And Computer Vision. He Has Supervised M.tech And B.tech Students. He Received Many Academic Related Awards. He Has Published A Book “Data Structures With C Programming” Arcler Press, 2010 Winston Park Drive Oakville Ontario L6h 5R7 Canada In 2019. He Has Published 12 Research Papers At National, International Conferences And International Journals. He Also Attended Several National & International Workshops.



# TABLE OF CONTENTS

---

<i>List of Figures</i> .....	<i>xi</i>
<i>List of Tables</i> .....	<i>xvii</i>
<i>List of Abbreviations</i> .....	<i>xix</i>
<i>Preface</i> .....	<i>xxi</i>
<b>Chapter 1    Introduction to Web Technology</b> .....	<b>1</b>
1.1. Concept of WWW .....	2
1.2. Concept of Internet .....	5
1.3. Http Protocol .....	7
1.4. Web Browser and Web Servers .....	13
1.5. Features of Web Versions .....	14
1.6. Concepts of Effective Web Design.....	16
1.7. Web Design Issues .....	18
1.8. Look and Feel of the Website .....	20
1.9. Page Layout and Linking .....	21
1.10 User Centric Design and Sitemap .....	22
1.11. Planning and Publishing Website .....	24
<b>Chapter 2    Basics of HTML</b> .....	<b>31</b>
2.1. Concept of HTML .....	32
2.2. Text Formatting and Fonts.....	34
2.3. Comment in Html Code, Color, and Hyperlink.....	39
2.4. Lists and Tables .....	46
2.5. Images and Forms .....	54
<b>Chapter 3    Introduction to CSS</b> .....	<b>61</b>
3.1. Style Sheets: Need of CSS .....	62
3.2. Basic Structure of CSS.....	64
3.3. How to Use CSS in Webpage.....	69

3.4. CSS Properties for Background Images and Colors .....	71
3.5. CSS Properties for Manipulating Texts and Fonts .....	78
3.6. Borders and Margins .....	88
3.7. Padding and Lists .....	95
3.8. Positioning in Css.....	101
3.9. Overview and Features of CSS2 and CSS3 .....	103
<b>Chapter 4 Basic of JavaScript.....</b>	<b>107</b>
4.1. Clients-Side Javascript .....	108
4.2. Data Types and Variables in Javascript.....	109
4.3. Functions in Javascript .....	112
4.4. Conditions Statement in Javascript .....	116
4.5. Javascript Loops .....	122
4.6. Javascript Pop Up Boxes .....	127
4.7. Advance Javascript: Javascript and Objects.....	132
4.8. DOM.....	138
4.9. Javascript form Validation.....	139
4.10. DHTML .....	141
<b>Chapter 5 Introduction to XML.....</b>	<b>149</b>
5.1. Uses of XML and Syntax of XML .....	150
5.2. XML DTD .....	154
5.3. XML Schema .....	158
5.4. Transforming XML Using XSL and XSLT.....	159
<b>Chapter 6 Introduction to PHP .....</b>	<b>165</b>
6.1. General Uses of PHP .....	166
6.2. How to Run PHP Script .....	166
6.3. Basic Syntax of PHP.....	168
6.4. Php Conditional or Decision Statements.....	170
6.5. Php Loops.....	172
6.6. Php and HTML, Arrays, and Functions.....	175
6.7. PHP Strings.....	180
6.8. Form Processing and Files.....	182

6.9. Cookies and Sessions.....	187
6.10. Basic Object Oriented Programing In PHP.....	194
<b>Chapter 7    PHP with MySQL .....</b>	<b>199</b>
7.1. Basic of MySQL.....	200
7.2. Create Table and Insert Data Using PHP .....	204
7.3. Altering Tables Using PHP.....	208
7.4. Update Data in a MYSQL Table .....	211
7.5. Delete Data and Tables From Database.....	213
7.6. PHP My Admin Features .....	217
<b>Chapter 8    Introduction Web to Python .....</b>	<b>219</b>
8.1. Advantages of Developing Web Applications in Python .....	220
8.2. Introduction to Python .....	221
8.3. Python with HTML .....	224
8.4. Submitting Forms with Python .....	226
8.5. Providing Text Area with Python .....	229
8.6. Checking Boxes with Python .....	230
8.7. Radio Buttons with Python.....	233
8.8. Drop Down (Select) Option with Python .....	235
8.9. Upload Files with Python.....	237
<b>Index.....</b>	<b>241</b>



# LIST OF FIGURES

---

**Figure 1.1.** WWW architecture

**Figure 1.2.** WWW works on client- server

**Figure 1.3.** The combination of various networks

**Figure 1.4.** Basic architecture of a web application and HTTP sites

**Figure 1.5.** UCD process

**Figure 2.1.** Output of firstpage.html

**Figure 2.2.** Output of bold.html

**Figure 2.3.** Output of italic.html

**Figure 2.4.** Output of highlight.html

**Figure 2.5.** Output of superscript.html

**Figure 2.6.** Output of small.html

**Figure 2.7.** Output of del.html

**Figure 2.8.** Output of ins.html

**Figure 2.9.** Output of comment.html

**Figure 2.10.** Output of background.html

**Figure 2.11.** Output of textcolor.html

**Figure 2.12.** Output of border.html

**Figure 2.13.** Output of colorvalue.html

**Figure 2.14.** Output of hyperlink.html

**Figure 2.15.** Output of imagehyperlink.html

**Figure 2.16.** Output of email.html

**Figure 2.17.** Output of ul\_list.html

**Figure 2.18.** Output of square\_list.html

**Figure 2.19.** Output of ol\_list.html

**Figure 2.20.** Output of upper\_list.html

**Figure 2.21.** Output of desc\_list.html

**Figure 2.22.** Output of nested\_list.html

**Figure 2.23.** Output of table.html

**Figure 2.24.** Output of tablehead.html  
**Figure 2.25.** Output of image\_web.html  
**Figure 2.26.** Output of image\_size.html  
**Figure 2.27.** Output of border\_image.html  
**Figure 2.28.** Output of align.html  
**Figure 2.29.** Output of form.html  
**Figure 2.30.** Output of form\_action.html  
**Figure 3.1.** Example of CSS 3.1  
**Figure 3.2.** CSS style rule  
**Figure 3.3.** Output of example 3.2  
**Figure 3.4.** Output of example 3.3  
**Figure 3.5.** Output of example 3.4  
**Figure 3.6.** Output of example 3.5  
**Figure 3.7.** Output of example 3.6  
**Figure 3.8.** Output of example 3.7  
**Figure 3.9.** Output of example 3.8  
**Figure 3.10.** Output of example 3.9  
**Figure 3.11.** Output of example 3.10  
**Figure 3.12.** Output of example 3.11  
**Figure 3.13.** Output of example 3.12  
**Figure 3.14.** Output of example 3.13  
**Figure 3.15.** Output of example 3.14  
**Figure 3.16.** Output of example 3.15  
**Figure 3.17.** Output of example 3.16  
**Figure 3.18.** Output of example 3.17  
**Figure 3.19.** Output of example 3.18  
**Figure 3.20.** Output of Example 3.19  
**Figure 3.21.** Output of example 3.20  
**Figure 3.22.** Output of example 3.21  
**Figure 3.23.** Output of example 3.22  
**Figure 3.24.** Output of Example 3.23  
**Figure 3.25.** Output of example 3.24  
**Figure 3.26.** Output of example 3.25  
**Figure 3.27.** Output of example 3.26

**Figure 3.28.** Output of example 3.27  
**Figure 3.29.** Output of example 3.28  
**Figure 3.30.** Output of example 3.29  
**Figure 3.31.** Output of example 3.30  
**Figure 3.32.** Output of example 3.31  
**Figure 3.33.** Output of example 3.32  
**Figure 3.34.** Output of example 3.33  
**Figure 3.35.** Output of example 3.34  
**Figure 3.36.** Output of example 3.35  
**Figure 4.1.** Output of call a function  
**Figure 4.2.** Output of parameters function  
**Figure 4.3.** Output of return function  
**Figure 4.4.** Output of if statement  
**Figure 4.5.** Output of if\_else statement  
**Figure 4.6.** Output of elseif statement  
**Figure 4.7.** Output of switch statement  
**Figure 4.8.** Output of for loop  
**Figure 4.9.** Output of for-in loop  
**Figure 4.10.** Output of for-of loop  
**Figure 4.11.** Output of while loop  
**Figure 4.12.** Output of do-while loop  
**Figure 4.13.** Output of alert popup  
**Figure 4.14.** Output of confirm popup  
**Figure 4.15.** Output of prompt popup  
**Figure 4.16.** Output of object  
**Figure 4.17.** Output of object  
**Figure 4.18.** Output of methods for object  
**Figure 4.19.** HTML DOM  
**Figure 4.20.** Output form validate  
**Figure 4.21.** Output example 4.20  
**Figure 4.22.** Output example 4.21  
**Figure 4.23.** Output JavaScript and HTML DOM 4.22  
**Figure 5.1.** Output of info.xml

**Figure 5.2.** Example of XSLT

**Figure 6.1.** Output of hello.php

**Figure 6.2.** Output of indexarray

**Figure 6.3.** Output of associativearray

**Figure 6.4.** Output of multiarray

**Figure 6.5.** Output of function

**Figure 6.6.** Output of parameter function

**Figure 6.7.** Output of htmlform

**Figure 6.8.** Output of welcome.php

**Figure 6.9.** Output of fopen

**Figure 6.10.** Output of writefile

**Figure 6.11.** Output of createcookie 6.11

**Figure 6.12.** Output of ModifyCookie 6.12

**Figure 6.13.** Output of DeleteCookie 6.13

**Figure 6.14.** Output of enabled 6.14

**Figure 6.15.** Output of demo\_session1.php 6.15

**Figure 6.16.** Output of demo\_session2.php 6.16

**Figure 7.1.** XAMPP control panel

**Figure 7.2.** PhpMyAdmin window

**Figure 7.3.** createdatabase.php

**Figure 7.4.** Newdb database in phpMyAdmin

**Figure 7.5.** Createtable.php

**Figure 7.6.** Student table inside the newdb database

**Figure 7.7.** insertdata.php

**Figure 7.8.** Insert values inside student table

**Figure 7.9.** Add new column inside student table

**Figure 7.10.** New column city inside student table

**Figure 7.11.** Drop column inside student table

**Figure 7.12.** Update row inside student table

**Figure 7.13.** Update Sweta to Vikram inside student table

**Figure 7.14.** Deleted record id=120 inside Student table

**Figure 7.15.** Student table in newdb database

**Figure 7.16.** Registration form data insert into student table

**Figure 7.17.** After submit button



**Figure 7.18.** Inside Student table inserted data

**Figure 8.1.** CGI architecture

**Figure 8.2.** Output hello.py

**Figure 8.3.** Output post.html

**Figure 8.4.** Output post.py

**Figure 8.5.** Output text.html

**Figure 8.6.** Output text.py

**Figure 8.7.** Output checkbox.html

**Figure 8.8.** Output checkbox.html

**Figure 8.9.** Output radiobutton.html

**Figure 8.10.** Output radiobutton.py

**Figure 8.11.** Output dropdown.html

**Figure 8.12.** Output dropdown.py

**Figure 8.13.** Output uploadfile.html

**Figure 8.14.** Output upload.py



# LIST OF TABLES

---

**Table 1.1.** The major differences between the web browser and web servers

**Table 1.2.** Difference between Web 1.0, Web 2.0 and Web 3.0

**Table 4.1.** A list of the reserved keywords in JavaScript

**Table 4.2.** A variety of methods of object

**Table 4.3.** Difference between HTML and DHTML

**Table 5.1.** Differences between HTML and XML



# LIST OF ABBREVIATIONS

---

API	Application Programing Interface
ARPANET	Advanced Research Projects Agency Network
CDN	Content Delivery Network
CGI	Common Gateway Interface
CSS2	Cascading Style Sheets Level 2
DHTML	Dynamic HTML
DNS	Domain Name Server
DOM	Document Object Model
DTP	Desktop Publishing
GMT	Greenwich Mean Time
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Number Authority
OOP	Object-Oriented Programing
OWL	Web Ontology Language
QBE	Query-by-Example
RDF	Resource Description Framework
RDFS	RDF Schema
SGML	Standard Generalized Markup Language
SPARQL	Simple Protocol and RDF Query Language
UCD	User-Centered Design
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web
XSD	XML Schema Definition
XSL	eXtensible Stylesheet Language



# PREFACE

---

The book highlights the fundamentals of web page development and prepares students to make real-world, industrial, or business strength web-based applications, and use a wide variety of Web development tools effectively and efficiently. Web technology is necessary for today because the internet has become the number one source of information, and many of the conventional software applications have become web applications. Web applications have become more strong and can fully replace desktop applications in most situations. Web technologies are the several tools and methods that are applied in the activity of communication between different types of devices over the internet. The basic principle, which will cover web browsers and some web app creation programming languages and frameworks which are used in the creation of websites. Data or information collected by the websites to store and access store data at the back-end is used databases connectivity programing. Hypertext markup language, referred to as HTML, is where the WWW started. It is a basic instruction that covers the basics concept of the web. CSS is one of the most important website design technologies that provide look and fill facilities for your web pages, and you can merge cascading style sheets (CSS) into your HTML code. JavaScript help to design interactive web pages, the involvement of graphical elements, database integration, and dynamics of your website. PHP stands for hypertext preprocessor. It is a scripting language preferred to use by both beginners and professionals in the web development areas.

This book reflects the author's experience in teaching courses on internet and web technology for more than 12 years. This book provides students and web developers with an understandable introduction to the web programming and scripting languages used to create Web sites and web applications. The main aim is to teach the programming concepts of different Web technologies and the fundamentals needed to program on the internet. We cover in this textbook fundamentals of internet, world wide web (WWW), HTML, CSS, JavaScript, XML, PHP, and connectivity with database using MYSQL and also cover the introduction of the web with python and CGI scripting to manage web from data. The contents of the book have been thoroughly organized and spread over eight chapters. Each chapter begins with the basics introduction and describes with syntax, code, result output, diagrams, and examples.

We trust this textbook would meet the rich practical hands-on experience in developing web applications, combined with teaching the subject for graduate/post-graduate students and who want to study the internet and web technology.

We would very much appreciate receiving any suggestions for improvement in the book from teachers, students, and other readers.

We wish to express our deep thanks to all those who helped in making this book a reality. We express our gratitude to our publisher and the team of publications that have taken great pains in publishing the book with speed and accuracy.



# CHAPTER 1

## INTRODUCTION TO WEB TECHNOLOGY

### CONTENTS

1.1. Concept of WWW .....	2
1.2. Concept of Internet.....	5
1.3. Http Protocol .....	7
1.4. Web Browser and Web Servers .....	13
1.5. Features of Web Versions .....	14
1.6. Concepts of Effective Web Design.....	16
1.7. Web Design Issues.....	18
1.8. Look and Feel of the Website.....	20
1.9. Page Layout and Linking .....	21
1.10 User Centric Design and Sitemap .....	22
1.11. Planning and Publishing Website.....	24

Web technology is a technique or method with the help of computers can exchange a few words or expressions with each other with the use of markup languages and multimedia are known as web technology. Web technology understands the basic concept of WWW, Internet that are an important component for this technology.

## 1.1. CONCEPT OF WWW

The World Wide Web (WWW) is an internet-supported service that uses a common set of rules recognized as protocols, to share out or exchange documents across the Internet in a standard manner. The web is a branch of the Internet; it can view through different available web browser software such as google chrome, internet explorer, Mozilla Firefox, etc. via browsers can access the digital libraries containing immeasurable articles, journals, images, e-books, news, tutorials stored in the form of web pages on computers around the world supported by web servers. The WWW is a structure that uses for exchanging information connecting computers via the Internet, bind them together into a vast collection of interactive multimedia software.

### 1.1.1. History of World Wide Web

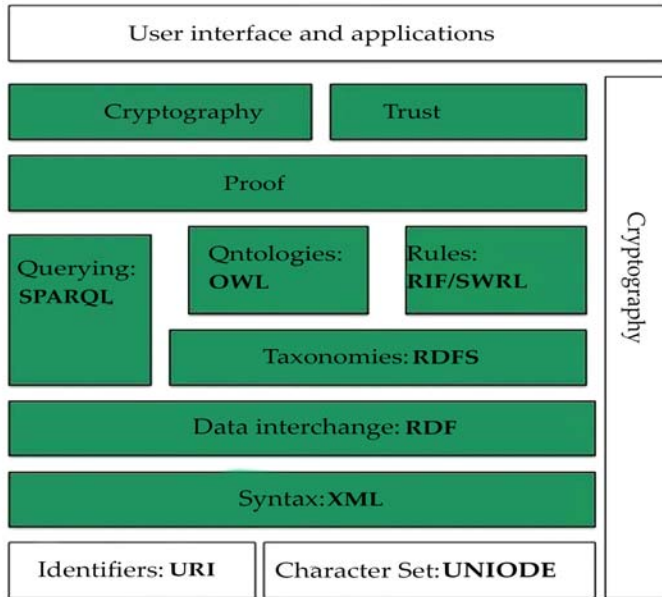
The WWW concepts were given by Tim Berners-Lee in 1989; In March 1989, Tim laid out his vision for the web in a file called “information management: A proposal.” Near October of 1990, Tim had given the three basic technologies that remain the foundation of today’s web technology:

1. **HTML:** Hypertext markup language that is markup (format) language for the web pages.
2. **URI:** Uniform resource identifier (URI) a type of “address” that is uniquely defined and used to identify to each resource on the web. It is also known as uniform resource locator (URL).
3. **HTTP:** Hypertext transfer protocol (HTTP) it allows for the retrieval of linked resources from across the web.

Near the end of 1990, the first web page was created. In April 1993, the WWW technology was available for everyone to use on a royalty-free basis. It has become the most powerful communication medium the digital world.

### 1.1.2. WWW Architecture

WWW architecture is divided into a number of layers as shown in the following Figure 1.1:



**Figure 1.1.** *WWW architecture.*

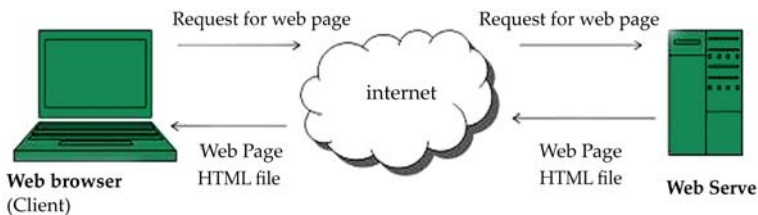
1. Identifiers and Character Set
  - **Uniform Resource Identifier (URI)** is used to uniquely identify resources on the web.
  - **UNICODE** makes it possible to build web pages that can be read and write in human languages.
2. Syntax
  - **XML (Extensible Markup Language)** assist to describe common syntax in semantic web.
3. Data Interchange
  - **Resource Description Framework (RDF)** framework assist to describe core representation of data for web. RDF represents data with reference to resource in the graph form.
  - **Taxonomies:** The RDF Schema (RDFS) is allowed to support standardized description of taxonomies and other ontological constructs.
  - **Ontologies:** The Web Ontology Language (OWL) proposes to build over RDFS. It comes in following three versions:
    - OWL Lite for taxonomies and simple constraints.

- OWL DL for full description logic support.
- OWL for more syntactic freedom of RDF.
- **Rules:** RIF and SWRL recommend rules beyond the constructs that are available from RDFs and OWL. Simple protocol and RDF query language (SPARQL) is SQL type language used for querying RDF data and OWL ontologies.
- **Proof:** All semantic and rules that are implemented at layer under the Proof and their result will be used to prove deduction.
- **Cryptography:** Cryptography means such as digital mark for verification of the beginning of sources is used.
- **User Interface and Applications:** At the top of layer a user interface and applications layer is built for users communication.

### 1.1.3. WWW OPERATION

The WWW mechanism works on client- server approach that shown in Figure 1.2. Following steps make clear how the web works:

1. Users enter the URL of the web page, in the address bar of the web browser.
2. Then browser send requests to the domain name server (DNS) for the IP address equivalent to URL (www.rgpv.ac.in.).
3. After receiving IP address, browser sends a request for web page to the web server via HTTP protocol which specifies the procedure between the browser and web server communicates.
4. After that web server receives request by HTTP protocol, checks it's and search for the requested web page. If found it returns back to the web browser and close the HTTP connection.
5. At this time the web browser receives the web page; it interprets it and displays the contents of web page in web browser's window.



**Figure 1.2.** *WWW works on client- server.*

## 1.2. CONCEPT OF INTERNET

The Internet is the combination of a variety of networks. We can access the internet services through any electronic device with a network connection like mobile phones and computers. It allows exchange of information or file between two or more computers that are connected on network. Therefore, internet helps in transfer of messages through mail, chat, video & audio conference, etc. It can be defined as following:

- Internet is a world-wide universal system of interconnected networks.
- Internet uses the standard internet protocol (TCP/IP).
- Every device or computer in internet is identified by a unique IP address, which identifies a computer location. A DNS is used to give name to the IP Address so that user can locate a computer by a name.
- Internet is easy to get every user all over the world.



**Figure 1.3.** *The combination of various networks.*

Internet concept was developed in 1969, under the project that known as advanced research projects agency network (ARPANET) to connect computers at different universities and U.S. defense. After some time people from different backgrounds such as engineers, scientists, students and researchers started by the network for exchanging information and messages. During 1990s the internet-working of ARPANET, NSFnet and other private

networks become into Internet. It includes millions of computing devices that carry and transfer volumes of information from one device to another.

### 1.2.1. Evolution

- The source of Internet invented from the concept of advanced research project agency network (ARPANET).
- ARPANET was developed by U.S. Defense Department.
- Basic purpose of ARPANET was to provide communication among the various organization of government.
- During 1972, the ARPANET extend over the globe with 23 nodes located at different countries and thus became known as Internet.
- By invention of new technologies such as TCP/IP protocols, DNS, WWW, browsers, scripting languages etc., Internet provided a standard to publish and access information over the web.

Internet are uses basic three principal:

- **Electronic mail:** Electronic mail, or e-mail, messages to users who have Internet E-mail addresses. Delivery time varies, but it's possible to send mail across the world and get a response in minutes.
- **USENET newsgroups:** USENET is a particular interest discussion groups, called newsgroups, to which readers can send or post messages which are distributed to other computers in the network. For example, alt.education.research, alt.education.distance, and misc.education.science.
- **Information files:** Government agencies, schools, and universities, commercial firms, interest groups, and private individuals place a variety of information on-line. The files were originally text only, but increasingly contain pictures and sound.

### 1.2.2. Difference between Internet and WWW

The Internet is also known as “interconnection of computer networks.” The internet is a huge network of networks. It connects millions of computers or electronic devices together worldwide, structure a network in which any computer can communicate with any other computer as long as they are both connected to the internet. The WWW, or just web, is a huge collection of digital pages or files to access information over the Internet. The web technology uses the HTTP protocol, for transmit and receive data, which

allows applications to communicate in order to exchange commercial motivation. The WWW also uses browsers, for example google chrome, internet explorer, Mozilla Firefox to access web documents called web pages that are linked to each other via hyperlinks. Web documents also contain multimedia (graphics, sounds, text and video) files.

### 1.3. HTTP PROTOCOL

Hypertext transfer protocol (HTTP) is standard and stateless protocol an application layer for distributed, hypermedia information system. HTTP is can be used for request and response methods, error codes, and headers. Mainly, HTTP is a TCP/IP based communication protocol, that is used to distribute data (like HTML files, image files, query results, etc.) on the WWW. By default TCP port is 80, but other ports can be used as well. It provides a standardized way how clients' request data and sent to the server, and how the servers respond to these clients' requests.

There are three basic features that make HTTP an easy and powerful protocol:

- **HTTP is connection less:** The HTTP client, a browser begins from an HTTP request and after a request is completed, the client waits for the response. The server processes the client request and sends a response back after which client disconnects the connection. Thus client and server know about each other for the duration of current request and response only. Additional requests are made on new connection similar to client and server is new to each other.
- **HTTP is media independent:** that means, any kind of data can be sent by HTTP as long as both the client and the server know how to handle the data content. It is required for the client as well as the server to specify the content type using suitable MIME-type.
- **HTTP is stateless:** that means, the server and client are aware of each other only during a current request. After that, both of them forget about each other, this nature of the protocol, neither the client nor the browser can maintain information between different requests across the web pages.

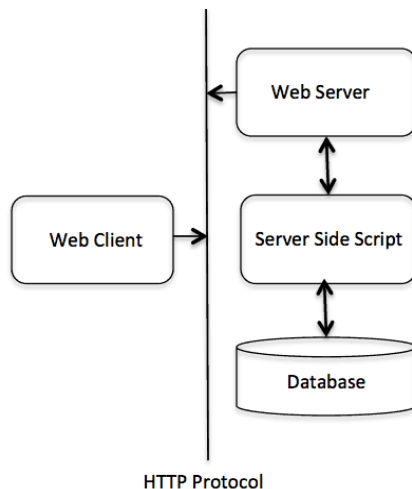
### 1.3.1. Fundamental Architecture of Web and HTTP

Figure 1.4 shows a basic architecture of a web application and show where HTTP sits: The HTTP protocol is a request or response protocol based on the client or server based architecture where web browsers and search engines, etc. act similar to HTTP clients and the web server acts as a server.

**Client:** HTTP client sends a request to the server in the form of a request method. That contain URI, and protocol version, pursue by a MIME-like message, request modifiers, client information, and possible body content over a TCP/IP connection.

**Server:** HTTP server responds with a status line, including the message's protocol version and a success or error code. That pursue by a MIME-like message containing server information, entity Meta information, and possible entity-body content.

**HTTP protocol parameters** and their syntax are used in the communication, such as, format for date, format of URL, etc. This will help you in create your request and response messages while writing HTTP client or server programs.



**Figure 1.4.** *Basic architecture of a web application and HTTP sites.*

### 1.3.2. HTTP Version

HTTP applies a <major>.<minor> numbering scheme to specify versions of the protocol. The HTTP version is indicated by an HTTP-Version field in the



first line. Here is the common syntax that specifying HTTP version number:

HTTP-Version = "HTTP" "/" 1\*DIGIT "." 1\*DIGIT

Example:

HTTP/1.0

or

HTTP/1.1

### 1.3.3. Uniform Resource Identifiers

Uniform Resource Identifiers (URI) is basically a formatted, case-insensitive string including name, location, etc. to identify a resource. A syntax of URI as follows:

URI = "http:" "/" host [ ":" port ] [ abs\_path [ "?" query ] ]

Here if the port is vacant or not given, port 80 is assumed for HTTP and an empty abs\_path is equivalent to "/".

Example: The following three URIs are equivalent:

http://xyz.com:80/~wsmith/home.html

http://XYZ.com/%7Ewsmith/home.html

http://XYZ.com:/%7ewsmith/home.html

### 1.3.4. Date/Time Formats

HTTP date/time stamps MUST be represented in Greenwich Mean Time (GMT), without any exception. HTTP is allowed to use any of the following three representations of date/time stamps:

Sun, 18 Dec 1994 08:49:37 GMT; RFC 822, updated by RFC 1123

Sunday, 18-Dec-94 08:49:37 GMT; RFC 850, obsoleted by RFC 1036

Sun Dec18 08:49:37 1994; ANSI C's asctime() format

### 1.3.5. Character Sets

Character sets to indicate the character sets that the client prefers. Multiple character sets can be listed and divided by commas. If a value is not specified, the default is the US-ASCII.

Example: the valid character sets:

US-ASCII

or

ISO-8859-1

or

ISO-8859-7

### 1.3.6. Content Encoding

A content encoding value indicates that an encoding algorithm has been used to encode the content before passing it over the network. Content coding is primarily used to allow a document to be compressed or otherwise usefully transformed without losing the identity. Every content-coding value is case-insensitive. Header HTTP/1.1 uses content-coding values in Accept-Encoding and Content-Encoding header fields.

Example: valid encoding schemes are:

Accept-encoding: gzip

Or

Accept-encoding: compress

### 1.3.7. Media Types

HTTP utilizes internet media types in the content-type and accept header fields in order to provide open and extensible data typing and type negotiation. The entire media-type values are registered with the internet assigned number authority (IANA). The general syntax of media type is as follows:

media-type = type "/" subtype \*(";" parameter)

Here type, subtype, and parameter attribute names are case-insensitive.

Example: Accept: image/gif.

- **HTTP Request:** The HTTP client, a browser begins from an HTTP request; HTTP clients send a request to a server in the form of a request message.
- **Request-Line:** The request-line begins with a method; followed by the request-URL and the protocol version, and ending with CRLF. The elements are separated by space SP characters.

Request-Line = Method SP Request-URI SP HTTP-Version

CRLF

### 1.3.8. Request Method

The request method indicates the process that performed on the resource identified by the given Request-URI. The method is case-sensitive and should be declaring in uppercase. The following lists supported methods in HTTP/1.1.

1. GET: GET method is used to retrieve information from the specified server using a given URI. GET method only retrieve data and should not effect on the other data.
2. HEAD: it transfers the status line and the header section only.
3. POST: POST method request is used to send data to the server, for example, client information, file upload, etc. using PHP or HTML forms.
4. PUT: Replaces all the present representations of the target resource with the uploaded content.
5. DELETE: Removes all the present representations of the target resource given by URI.
6. CONNECT: Create a subway to the server identified by a given URI.
7. OPTIONS: Express the communication options for the target resource.
8. TRACE: Performs a message loop back test along with the path to the target resource.

### 1.3.9. Request-URI

The Request-URI is a URI that identifies the resource upon which to apply the request. Following are generally used to specify an URI:

Request-URI = “\*” | absoluteURI | abs\_path | authority

- The asterisk \* is used when an HTTP request does not apply to a particular resource, but to the server itself, and is only allowed when the method used. It does not necessarily apply to a resource. For example: OPTIONS \* HTTP/1.1
- The absolute URI is used when an HTTP request is being ready to a proxy. The proxy is requested to forward the request or service from a valid cache, and return the response. For example:

GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1

- The Request-URI is used to identify a resource on a source server or gateway. For example, a client wishing to retrieve a resource directly from the origin server would create a TCP connection to port 80 of the host “www.ibc.org” and send the following lines:  
GET /pub/WWW/Project.html HTTP/1.1  
Host: www.ibc.org  
Note absolute path cannot be blank; if none is there in the original URI, it MUST be given as “/” (the server root).
- **HTTP response:** After receiving and interpreting a request message from HTTP client, a server responds with an HTTP response message.
- **Message Status-Line:** Message Status-Line consists of the protocol version followed by a numeric status code and it's linked with textual phrase. The elements are separated by space SP characters.

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

- **HTTP Version:** A server supporting HTTP version 1.1 and it will return version information:  
HTTP-Version = HTTP/1.1
- **Status Code:** The Status-Code element is a 3-digit figure where first digit of the Status-Code defines the class of response and the last two digits do not have any categorization role. There are 5 standards for the first digit:
  1. 1xx: Informational: that means the request received and the process is continuing.
  2. 2xx: Success: that means the action was successfully received, understood, and accepted.
  3. 3xx: Redirection: that means further action must be taken in order to complete request.
  4. 4xx: Client Error: that means the request contains incorrect syntax or cannot be fulfilled.
  5. 5xx: Server Error: that means the server failed to fulfill an actual valid request.

## 1.4. WEB BROWSER AND WEB SERVERS

In favor of internet communication, we require a web browser and web servers. The client or web browser sends requests for web documents or services. The message that goes through the web browser to the webserver is known as an HTTP request. When the webserver receives the request, it searches and stores to find the appropriate page. If the webserver is able to locate the page, it posts up to the HTML contained within (using transport layer protocol), addresses these post to the browser (using HTTP), and transmit them back across the network.

**Table 1.1.** The major differences between the web browser and web servers

WEB BROWSER	WEB SERVER
Web browser is an application program that shows a WWW document. It generally uses the internet service to access the document.	Web server is a program or the computer that provides services to other computer application program called client.
The web browser sends requests the server for the web documents and services.	The web server accepts request, approve and respond to the request send by the web browser for a web document or services.
The web browser acts as an interface between the server and the client and shows a web document to the web browser.	The web server is software or a system which maintains the web applications, make response and accept client's data.
The web browsers send an HTTP request and get an HTTP response.	The web server gets HTTP requests and sends HTTP responses.
The web browsers doesn't exist any processing model for the web browser.	The web server exist three types of processing models for web server, i.e., Process-based, Thread based and Hybrid.
The Web browser stores the cookies for different websites.	The Web servers provide an area to store and arrange the pages of the website.
The web browser is installed on the client-side computer.	The web server can be a remote machine located at the other side of your network or even on the other end of the globe.

If the webserver is unable to find the requested page, it sends a page containing an error message (i.e., Error 404 – page not found), and it posts up to dispatches that page to the browser. This message received by the web browser by the server is called the HTTP response.

## 1.5. FEATURES OF WEB VERSIONS

1. **Web 1.0:** The first stage of the WWW development refers to web 1.0. Previously, there were only few content creators in Web 1.0 with the enormous majority of users who are consumers of content. Private web pages were common, consisting mainly of static pages hosted on ISP-run web servers, or on free web hosting services. Web 1.0 is a content delivery network (CDN) which enables to showcase the piece of information on the websites. It can be used as personal websites. It costs to user as per pages viewed.
  - i. Four essentials design of a Web 1.0 site:
    - a. It has Static pages;
    - b. Web pages are served from the server's file-system;
    - c. Pages built using Server Side Includes or common gateway interface (CGI); and
    - d. Frames and Tables used to arrangement and align the elements on a page.
2. **Web 2.0:** The web 2.0 refers to world wide website which highlights user-created content, usability and interoperability for end users. Web 2.0 is also named as participative social web. It does not refer for any amendment to any technical specification but to modify in the way, Web pages are designed and used. An interaction and collaboration with each other is allowed by Web 2.0 in a social media conversation as designer of user-generated content in a virtual community. Web 2.0 is improved version of Web 1.0. The web browser technologies are used in Web 2.0 expansion and it includes AJAX and JavaScript frameworks. In recent times, AJAX and JavaScript frameworks have become a very popular way of creating web 2.0 sites.
  - i. Five major features of Web 2.0:
    - a. Open sorting of information, permits users to retrieve and classify the information jointly.

- b. Dynamic content that is quick to respond to user input.
  - c. Information flows between owner site and user's site that means of assessment & online commenting.
  - d. Developed APIs to permit self-usage, for example by a software application.
  - e. Web access guide to concern different, from the conventional Internet user to a wider variety of users.
3. **Web 3.0:** Web 3.0 refers the development of web utilization and interaction which includes altering the web into a database. It enables the up gradation of back-end of the web, and focus on the front-end (Web 2.0 has mostly been about AJAX, tagging, and another front-end user-experience innovation). Web 3.0 is a term which is used to describe many evolutions of web usage and interaction among several paths. The semantic web (3.0) establishes "the world's information" in more reasonable way than google can ever attain with their existing engine schema. This is mainly true from the perspective of machine conception as opposed to human understanding. The semantic web uses a declarative ontological language like OWL to produce domain-specific ontologies that machines can use to reason about information and make new conclusions.
- i. Five main features of Web 3.0
- a. **Semantic web:** The semantic web improves web technologies in demand to create, share and connect content through search and analysis based on the capability the meaning of words, rather than on keywords or numbers.
  - b. **Artificial intelligence:** Combining this capability with natural language processing, in Web 3.0, computers can distinguish information like humans in order to provide faster and more relevant results. They become more intelligent to fulfill the requirements of users.
  - c. **3D graphics:** The three-dimensional design is being used widely in websites and services in Web 3.0. Museum guides, computer games, e-commerce, geo-spatial contexts, etc. are all examples that use 3D graphics.
  - d. **Connectivity:** With Web 3.0, information is more connected thanks to semantic metadata. As a result, the user experience

evolves to another level of connectivity that leverages all the available information.

- e. **Ubiquity:** Content is accessible by multiple applications, every device is connected to the web, the services can be used everywhere.

**Table 1.2.** Difference between Web 1.0, Web 2.0 and Web 3.0

WEB 1.0	WEB 2.0	WEB 3.0
Mostly read-only	Wildly read-write	Portable and personal
Company focus	Community focus	Individual focus
Home pages	Blogs / wikis	Live-streams / waves
Owning content	Sharing content	Consolidating content
Web forms	Web applications	Smart applications
Directories	Tagging	User behavior
Page views	Cost per click	User engagement
HTML/Portals	XML / RSS	RDF / RDFS / OWL

## 1.6. CONCEPTS OF EFFECTIVE WEB DESIGN

1. **Web designing:** The term ‘web design’ may also point to the visual aspect of a website. Design process not only includes front end designing but also the process of writing the markup. Web design is a concept of planning, creating, and maintaining the websites. as well the creation and updating, this concept also involves take care of the user interface, the architecture of information present, the layout, the colors, content, navigation ergonomics, and designs of the various icons.

### 1.6.1. Effective Web Design Principles

1. **Purpose:** Good web design always provides by requirements of the user and web visitors looking for information, entertainment, type of interaction, or to transact with your business. All pages of your website needs to have a clear purpose, and to fulfill a specific need for your website.



2. **Communication:** Users on the web have a tendency to want information quickly, so it is important to communicate clearly, and make your information easy to read and grasp. Some effective strategy to include in your web design include: organizing information using headlines and sub headlines, using bullet points instead of long sentences.
3. **Typefaces:** In common, Sans Serif fonts such as Arial and Verdana are easier to read online. The ideal font size for reading without difficulty, online is 16px and stick to a maximum of 3 typefaces in a maximum of 3 point sizes to keep your design streamlined.
4. **Colors:** A well idea out colors palette can go a long way to improve the user experience. Balancing of colors create stability and harmony. Using contrasting colors for the text and background will make reading or looking easier for the eye. Animated colors create emotion and should be used carefully. White space/negative space is very effective at giving your website a modern and smart look.
5. **Images:** An image can speak a thousand words, and choosing the correct images for your website can help with brand positioning and connecting with your target viewers. Also using infographics, videos and graphics as these can be much more effective at communicating than well written section of text.
6. **Navigation:** Navigation is concerning how it is easy for people to take action and move around your website. Some strategy for effective navigation includes a logical page hierarchy, using bread crumbs, designing clickable buttons.
7. **Grid based layouts:** Placing content randomly on your web page can end up with a random appearance that is confused. A grid base layout arrange content into sections, columns and boxes that line up and feel balanced, which leads to a better looking website design.
8. **“F” pattern design:** Eye tracking studies have acknowledged that people scan by computer screens in an “F” pattern. Most of what people see is in the top and left of the screen and the right side of the screen are rarely seen. Effectively designed websites will work with a reader’s natural behavior and show information in order of importance (left to right, and top to bottom).

## 1.7. WEB DESIGN ISSUES

1. **Most important technical issues in web design:** Before you make a website, you should consider the technical issues relating to web design, particularly:
  - Browser compatibility;
  - Screen resolutions;
  - Web technologies; and
  - Internet speed.
2. **Browser compatibility:** Web pages should be able to show across different browsers, like Internet Explorer, Firefox, Safari and Chrome etc. When creating your site, test your web pages for browser compatibility issues in as many browsers and operating systems as you can. Keep in mind to test on most recent browser versions, as well as the older versions, not all of your visitors may be using up-to-date software. If you are updating an existing site, use web analytic tools to observe what browsers your customers are currently using to access your website.
3. **Screen resolutions:** The most common screen resolution size in recently has been:
  - 1366 x 768 pixels for desktops
  - 360 x 640 pixels for mobile screens
  - 768 x 1024 pixels for tablets
4. **Internet speeds:** Not each and every internet users have high-speed access, so connection speed should also influence your webpage design. Study suggests that:
  - almost half of web users expect a webpage to load in 2 seconds or less
  - 40 per cent of people abandon a website that takes more than 3 seconds to load

Too many images or rich media - such as animations or video - will slow down the speed at which your webpage loads. This can result in your customers leaving the site. Since page speed is a ranking factor, slow speeds can also hurt your search ranking. Try to keep file and image sizes to a minimum. The total size of a webpage should be no more than 40 to 60 kilobytes.

5. **Technology:** Some web technologies can prevent users from viewing your site or affect indexing of your website by search engines. These include:
  - HTML frames
  - JavaScript
  - Flash
  - AJAX

If using any of these technologies, consider the potential risks to the usability and accessibility of your website. See more on web accessibility issues and learn how to design a user-friendly website.

### 1.7.1. Some of the Most Common Problems in Website Design

1. **There's no clear path:** You want to extend a warm welcome to your visitors. Give them an easy way in and through. Too much competition for attention is a turn off. When you provide too many options, the functional result is no options.
2. **Outdated design:** Your site was state of the art in 2009. It's got a header, a couple of sidebars, and a big chunk of information running down the center. Guess what? It looks like it's nearly a decade old. Because it is. Time to refresh with current design thinking. The layout of a page has evolved over the past decade. These days the best sites break up content into smaller, digestible bits.
3. **Overused stock images and icons:** If visitors see the same image on multiple sites, it erodes trust. That picture of people sitting around the conference table? They sure get around to a lot of offices!
4. **Too many textures and colors:** You are trying to add interest, but you just add clutter. Limit colors and fonts. Maintain a thematic color scheme. For professional sites, try to limit the variety of fonts to three or fewer.
5. **Design for the wrong reasons:** Always begin by identifying your target audience and customizing design and content. You may want your site to look "modern" or like another site you've seen, but if you haven't checked in with what your audience needs and wants, you can fail miserably.

6. **Cute that doesn't cut it:** When your links have adorable, witty names, the experience gets tired fast. Links that don't make much sense are not user friendly and won't ingratiate you with your visitors. Be practical and basic when naming links. Make it easy for people. Design for multiple visits. A rotating banner is cool the first time, and maybe the second, but at some point it's just a stale eyesore. Monotonous calls to action.
7. **Your site isn't optimized for mobile:** You shouldn't need to be reminded of this, but numbers don't lie. Mobile is overtaking desktop. It's increasingly likely that your visitors see your site on a tiny screen. If they have to pinch and stretch to read, they'll find a better source of information. Be sure to test your site on smartphone and tablet.
8. **You play hard to get:** If you want customers to find you, make sure your address, phone number and hours of operation are easily accessible on your site. Too often, that information is hidden or completely absent.

## 1.8. LOOK AND FEEL OF THE WEBSITE

The essential terms, “look and feel” of a website is how the site looks to the user and how it feels when he or she is interacting or visiting with it.

- The “look” is defined by the following components of your website:
  - Color palette;
  - Images;
  - Layout;
  - Font choices; and
  - Overall styling;
- The “feel” is determined by these characteristics:
  - The movement and response of dynamic components like dropdown menus, buttons, forms, and galleries;
  - Sound effects; and
  - The speed by which pages and images load.

### **1.8.1. Why is the Look and Feel of a Website Important?**

Website in general look and feel is important because it immediately conveys an attitude to your clients before they even start reading the content on the site. Before you start a website redesign, check your goals against business standards by looking at your competitors' websites. A website should look clean, powerful and well organized. A website for a group or fashion designer can be more creative with colors, texture and image choices. The look and feel of a website can also be described as the website's "qualities."

### **1.8.2. How to Use "Look and Feel" to Enhance Your Web Design**

Look and feel can be describing using adjectives just like you would describe a friend or business associate. By using precise adjectives, you can assist the team at your chosen web design company in their layout and design choices before they present their work to you. Here are several examples of the types of adjectives you might use to describe your website:

- Friendly;
- Approachable;
- Professional;
- Experienced;
- Upscale;
- Exclusive;
- Cutting edge;
- Stylish; and
- High-tech.

## **1.9. PAGE LAYOUT AND LINKING**

### **1.9.1. Page Layout**

Page layout refers to the arrangement of text, images, and additional objects on a page. This word was firstly used in desktop publishing (DTP), but is now commonly used to describe the layout of web pages also. Page layout techniques are used to make especially for the appearance of magazines, newspapers, books, websites, and other types of publications. Page layout is the branch of graphic design that deals in the arrangement of visual elements

on a page. The page layout of a printed or electronic document encompasses all elements of the page. This includes the page margins, text blocks, images, object padding, and any grids or templates used to define positions of objects on the page. Page layout applications, such as Adobe InDesign and QuarkXPress, allow page designers to modify all of these elements for a printed publication.

### 1.9.2. Page Link

A link (short for hyperlink) is an HTML object that allows you to jump or migrate to a new location when you click or tap it. Links are found on almost every webpage and provide a simple means of navigating between pages on the web. Links can be attached to text, images, or other HTML elements. Most text links are blue; since that is standard color web browsers use to display links. However, links can be any color since the style of the link text may be customized using HTML or CSS styles. In the early days of the web, links were underlined by default. Nowadays, underlining links is less common.

When a link is applied to an image, the link tag encapsulates, or surrounds the image tag. Since the image tag is nested inside the link tag, the image itself becomes a link. This method can be used to apply links to other elements such as `<div>` and `<span>` objects. However, since CSS can be used to stylize a link, an `<a>` tag with a CSS class or ID attribute is often used in place of a `<div>` or `<span>` tag.

Example of the HTML for a text and image link:

- Text Link

```
<a href="/definition/computer">Computer Definition</a>
```

- Image Link:

```
<a href="https://techterms.com/definition/computer"></a>
```

## 1.10 USER CENTRIC DESIGN AND SITEMAP

### 1.10.1. Concept of User-Centered Design (UCD)

User-centered design (UCD) is an iterative design procedure in which designers focus on the users and their requirements in each phase of the design procedure. In UCD, design teams engage users throughout the design procedure via a variety of research and design techniques, to create

extremely usable and accessible products for them. The following are the general phases of the UCD process:



**Figure 1.5.** *UCD process.*

- Identify the context of use: Identify the people who will use the product, what they will use it for, and under what conditions they will use it.
- Identify requirements: Identify any business requirements or user goals that must be met for the product to be successful.
- Create design solutions: This part of the process may be done in stages, building from a rough concept to a complete design.
- Evaluate designs: Evaluation - ideally through usability testing with actual users - is as integral as quality testing is to good software development.

### 1.10.2. User-Centered Design Principles

There are five major UCD principles:

- A clear understanding of user and task requirements.
- Incorporating user feedback to define requirements and design.

- Early and active involvement of the user to evaluate the design of the product.
- Integrating UCD with other development activities.
- Iterative design process.

The essential elements of UCD:

- **Visibility:** Users should be able to see from the beginning what they can do with the product, what it is about, how they can use it.
- **Accessibility:** Users should be able to find information easily and quickly. They should be offered various ways to find information for example call to action buttons, search option, menu, etc.
- **Legibility:** Text should be easy to read. As simple as that.
- **Language:** Short sentences are preferred here. The easier the phrase and the words, the better.

### 1.10.2. Sitemaps

A sitemap is a blueprint of your website that help search engines find, crawl and index all of your website's content. Sitemaps also tell search engines which pages on your site are most important.

There are four main types of sitemaps:

- **Normal XML sitemap:** This by far the most common type of sitemap. It's usually in the form of an XML Sitemap that links to different pages on your website.
- **Video sitemap:** Used specifically to help Google understand video content on your page.
- **News sitemap:** Helps Google find content on sites that are approved for Google News.
- **Image sitemap:** Helps Google find all of the images hosted on your site.

## 1.11. PLANNING AND PUBLISHING WEBSITE

### 1.11.1. Planning a Website

1. **Set your purpose and goals:** What is the purpose of your website, gain publicity for your business, sell your inventory, rally support



behind a cause? It's important to identify your website's purpose, as well as your target audience. You should also define your goals. Set measurable, specific goals for your website that is in line with your marketing goals. An analytic tool like Google Analytic will allow you to monitor your website's performance over time.

2. **Create a budget:** Whether you're an established, mid-sized organization or a fledgling start-up, you should always set a budget for your website expenses. This will probably include funds for web design, programing, and web hosting. Research the market by shopping around and consulting with professionals. It's better to choose team members based on experience, references, and examples of work.

3. **Assign roles:** Company stakeholders (owner, marketing manager, or whoever else represents a primary function of the business)

- Web developer
- Content writer and/or editor
- HTML/CSS professional
- Web and graphic designer

Make sure everyone on your team knows their role and what is expected of them, and that they stay abreast of deadlines and new developments.

4. **Create a content strategy:** What kind of content will you be displaying on your website? Content is basically anything that gives your visitors information. It can include, but is not limited to:

- Blog posts;
- Documents;
- Video;
- Pictures (such as in a gallery); and
- Slideshows.

Embedded social media feeds (for example Twitter stream or Face book page updates) your content strategy is the way that you plan to present your content over time. Since content is such a vital aspect of a website, bring in help if you need it. Hire a writer who is experienced with writing for the web, and invest in some professional looking pictures of your storefront and employees.

5. **Structure your website:** Decide what pages you'll be using and what features will be on each one. Most websites have an about and Contact page, but the pages you use should meet your business' needs.
6. **Start designing:** The importance of good web design can't be stressed enough. Good website design includes both usability and esthetics. An ugly website will drive away visitors, as will a website that's difficult to navigate. Keep in mind some basic concepts of usability as you go:
  - Make your navigation easy to understand and easy to find. Research shows that most users expect website navigation to be vertical and centered at the top of the page.
  - Use an easy-to-read font for blocks of text. Choose a background color and text color that contrast well (Hint: No red text on a hot pink background).
  - Make sure your site fits the screen. Use responsive design (or an equally effective approach) to make your website one that adapts to all screen sizes.
  - Keep your website light so that it loads quickly.
  - Make the company logo and tag line prominent on the page.
  - Keep styles and colors consistent across the website.
  - Make copy clear and concise, and put important information and features (e.g., your newsletter sign-up form) above the fold.
  - Make notes about what to include in the style sheet as you design, as you want to keep style and function separate. This is important, not only to comply with web standards, but to make it easier to change something in the future if you need to.
  - You should also design with the future in mind. For instance, your website may only have a few blog posts now, but what about when you have two hundred?
7. **Test it out:** Testing is important for getting out bugs out and catching details that you might have missed initially. Make sure your website shows up the way you want it to in all browsers, including Chrome, Firefox, Internet Explorer, and mobile web browsers like Safari and Opera Mini. Test it on your cell phone,

your tablet, and your colleague's cell phones and tablets too. You want your site to have a consistent appearance no matter what screen it shows up on. Make sure all of the links work, that the images are properly sized, and that you've replaced all of the placeholders with actual content. See to it that all of the forms and other input fields are working.

8. **Maintain your site:** Once your site is launched, the work isn't over. A website is an ongoing entity that continuously represents your company, so maintenance is very important. Monitor your analytic software to see how your website is performing with the public. Keep an eye on metrics like your number of unique visitors, bounce rate, and which pages are most popular on your website. You might find that certain metrics are more useful to you than others, but that is information you'll find out over time.

### 1.11.2. Publishing Website

Web publishing is the procedure of publishing original content on the Internet. The process includes building and uploading websites, updating the associated webpages, and posting content to these webpages online. Web publishing comprises of personal, business, and community websites in addition to e-books and blogs. The content meant for web publishing can include text, videos, digital images, artwork, and other forms of media. Publishers must possess a web server, web publishing software, and an Internet connection to carry out web publishing. Web publishing is also known as online publishing.

A publisher requires three things to publish content on the Internet:

- Website development software;
- Internet connection; and
- A web server to host the website.

The website development software can be a professional web design application like dream-weaver or a straightforward web-based content management system like Word Press. Publishers require an internet connection to upload the content to the web server. Major sites may utilize a dedicated server to host them; however, many smaller websites usually reside on shared servers that host an array of websites.

## 1.12. DESIGNING EFFECTIVE NAVIGATION

### 1.12.1. Navigation

That facilitates movement from one Web page to another Web page. Navigation is often taken for granted, but it plays a crucial role in getting site visitors to view more than just the home page. If navigation choices are unclear, visitors may elect to hit the “Back” button on their first (and final) visit to a Web site. Once they enter, the real challenge begins, as it is no easy task to allow first-time visitors to get takes maximum advantage of a site.

### 1.12.2. Basic Rules to Create Effective Website Navigation

1. **Keep it consistent:** Nothing is more irritating than a site navigation that changes as we move from one section of the website to the next. You should aim at establishing and maintaining visitors’ trust in your website. This trust is essential in increasing your website conversion. What enhances trust for one visitor versus the next varies tremendously, which is also why creating personas for your website is so crucial.  
To enhance visitors’ trust on your website, you will find six areas to focus on:
  - Value proposition;
  - Design aspects;
  - Continuity;
  - External reputation;
  - Social proof; and
  - Membership/professional organization or affiliation.
2. **Keep it simple:** You do not have to get too fancy when designing your website navigation. Straightforward navigation wins over complex navigation anytime.
3. **Orient your visitors:** Only 20% of visitors are in the action buying stage. These visitors are typically highly motivated and know what they are looking for. They have decided to buy, and if you present them with the right information, you can convert them. Another 20% to 30% of your site visitors landed there by mistake. There is little you can do about these visitors. The remaining 50% are still early in the buying funnel. This includes visitors who are

in the awareness stage, those who are researching, and those who are evaluating alternatives. Your website navigation must meet the needs of visitors in all different stages of the buying funnel. For each buying stage, your website should orient visitors on where they are and where to go to next.

4. **Match visitors' expectation:** If visitors are expecting to see certain elements, make sure that you anticipate and display them. Site navigation must match the visitor cognitive progression. Cognitive progression focuses on the information we predict a visitor needs to in order for him to convert.
5. **Match visitors' language:** If your visitors refer to a product feature using a specific term, then forget about using manufacturer or industry terminologies. You should talk directly to the visitors who are surfing your website and whom you are trying to convert. This, of course, will be challenging if different segments of your website visitors use different terms for the same feature. We see this lot with websites that serve customers from different parts of the world.
6. **Test your website navigation:** The approach to testing your website design varies if you are working with a new website or if you are dealing with an existing one.



## CHAPTER 2

# BASICS OF HTML

### CONTENTS

2.1. Concept of HTML .....	32
2.2. Text Formatting and Fonts.....	34
2.3. Comment in Html Code, Color, and Hyperlink .....	39
2.4. Lists and Tables .....	46
2.5. Images and Forms .....	54

In the web programming language, HTML manuscript is supposed to be the structural part of a web page. It is the base layer where an instruction should be delivered or displayed and the behavioral layer scripting and interactivity is applied.

## 2.1. CONCEPT OF HTML

Hypertext markup language refers to as HTML. It is a markup language not a programming language that defines the structure of your web content. HTML consists of a sequence of tags or elements, which you use to enclose or wrap different parts of the content to make it interactivity. The enclosing tags can make font bigger or smaller, italicize words, make a text or image hyperlink to anywhere else, and so on. The basic HTML tags which divides the complete document into various parts like head, title, body etc.

- All HTML document begins with a HTML tag, while this is not compulsory but it is a good convention to start the document with this under mentioned tag:  
`<!DOCTYPE html>`
- **<html>**: Every HTML code should be enclosed between basic HTML tags. It begins with `<html>` and ends with `</html>` tag.
- **<head>**: The head tag comes which contains all the header information of the web page or document like the title of the page and other miscellaneous information. This information is enclosed within head tag which opens with `<head>` and ends with `</head>`.
- **<title>**: The title of a web page mentioned by `<title>` tag, this is header information and hence mentioned within the header tags. The tag begins with `<title>` and ends with `</title>`.
- **<body>**: The body tag enclosed actual body of the web page which will be visible to all the users. This tag begins with `<body>` and ends with `</body>`.
- **HTML headings**: These tags help us to give headings of webpage content and these tags are generally written inside the body tag. HTML provides six heading tags from `<h1>` to `<h6>`. Every tag displays the heading in a special style and font size.
- **HTML paragraph**: These tags help us to write paragraph statements in a webpage. This tag begins with the `<p>` tag and



ends with `</p>` and break or new line used to the `<br>` tag, `<br>` is an empty tag.

- **Horizontal lines:** The `<hr>` tag is used to break the page into different parts, by creating horizontal margins from left to right hand side of the web page. `<hr>` is also an empty tag and doesn't take any other statements.

### Example: firstpage.html 2.1

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Run this HTML Code using following steps:

- Write or copy the following HTML code into notepad.
- Save the file on your computer. Select **File > Save as** in the notepad menu.
- Name the file using .html or .htm extension Ex: **“index.html”**
- Open the saved HTML file in your preferred browser (double clicks on the file, or right-click – and choose “Open with”).

Output: firstpage.html



**Figure 2.1.** *Output of firstpage.html.*

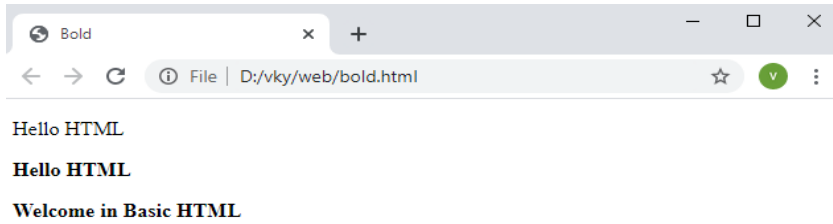
## 2.2. TEXT FORMATTING AND FONTS

- **HTML Formatting Elements:** Formatting elements were expected to display particular types of text:
  - `<b>` – Bold text
  - `<strong>` – Important text
  - `<i>` – Italic text
  - `<em>` – Emphasized text
  - `<mark>` – Marked text
  - `<small>` – Smaller text
  - `<del>` – Deleted text
  - `<ins>` – Inserted text
  - `<sup>` – Superscript text
1. **Making text bold or strong:** This tag begins with `<b>` and close with `</b>` tag, this tag help to make the text **bold**. We can also use the `<strong>` tag to make the text strong, It also opens with `<strong>` and ends with `</strong>` tag.

Example: bold.html 2.2

```
<!DOCTYPE html>
<html>
<head>
<title>Bold</title>
</head>
<body>
<p>Hello HTML</p>
<p><b> Hello HTML </b></p>
<p><strong>Welcomein Basic HTML </strong></p>
</body>
</html>
```

Output:



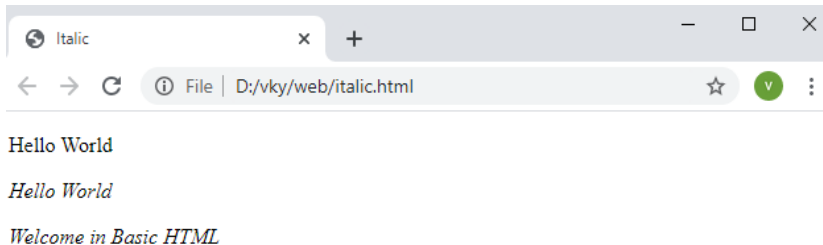
**Figure 2.2.** *Output of bold.html.*

- 2. Making text Italic or emphasize:** The `<i>` tag is used to *italicize* the text, it starts with `<i>` and ends with `</i>` tag. The `<em>` tag is used to *emphasize* the text, it starts with `<em>` and ends with `</em>` tag.

### Example: italic.html 2.3

```
<!DOCTYPE html>
<html>
<head>
<title>Italic</title>
</head>
<body>
<p>Hello World</p>
<p><i> Hello World </i></p>
<p><em>Welcome in Basic HTML</em></p>
</body>
</html>
```

### Output:



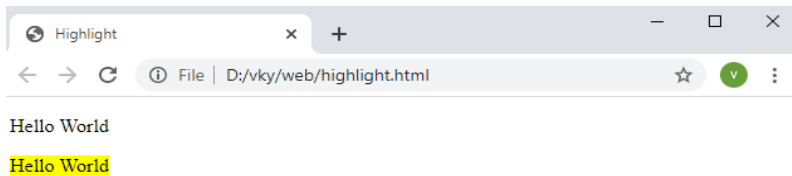
**Figure 2.3.** *Output of italic.html.*

3. **Highlighting a text:** It is highlight a text in HTML using the `<mark>` tag, It has an opening tag `<mark>` and a closing tag `</mark>`.

**Example: highlight. html 2.4**

```
<!DOCTYPE html>
<html>
<head>
<title>Highlight</title>
</head>
<body>
<p> Hello World </p>
<p><mark> Hello World </mark></p>
</body>
</html>
```

Output:



**Figure 2.4.** *Output of highlight.html.*

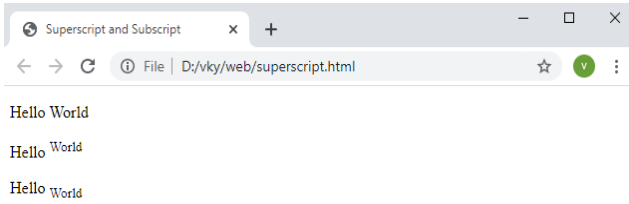
4. **Making a text Subscript or Superscript:** The `<sup>` element is used to superscript a text and `<sub>` element is used to subscript a text. It has an opening tag `<sup>` and a closing tag `</sup>`, as `<sub>` and `</sub>`.

**Example: superscript.html 2.5**

```
<html>
<head>
<title>Superscript and Subscript</title>
</head>
<body>
```

```
<p> Hello World </p>
<p>Hello <sup>World </sup></p>
<p>Hello <sub> World </sub></p>
</body>
</html>
```

Output:



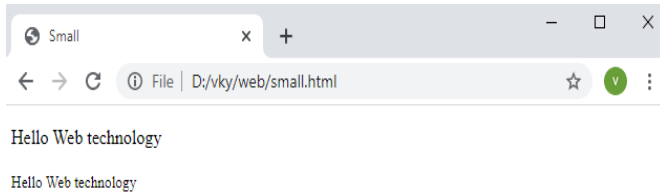
**Figure 2.5.** *Output of superscript.html.*

- 5. Making text smaller:** The `<small>` element is used to make the text smaller. The text that needs to be displayed smaller should be written inside `<small>` and `</small>` tag.

#### **Example: small.html 2.6**

```
<!DOCTYPE html>
<html>
<head>
<title>Small</title>
</head>
<body>
<p>Hello Web technology</p>
<p><small>Hello Web technology</small></p>
</body>
</html>
```

**Output:**



**Figure 2.6.** *Output of small.html.*

- 6. HTML <del>:** <del> element defines text that has been deleted from a document.

**Example: del.html 2.7**

```
<!DOCTYPE html>
<html>
<head>
<title>Deleted</title>
</head>
<body>
<p>My favorite color is <del>blue</del> red.</p>
</body>
</html>
```

**Output:**



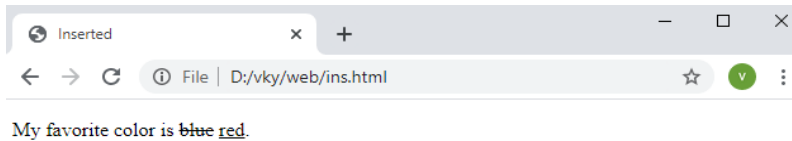
**Figure 2.7.** *Output of del.html.*

7. **HTML <ins>:** The HTML <ins> element defines a text that has been inserted into a document.

**Example: ins.html 2.8**

```
<!DOCTYPE html>
<html>
<head>
<title>Inserted</title>
</head>
<body>
<p>My favorite color is <del>blue</del><ins>red</ins>.</p>
</body>
</html>
```

**Output:**



**Figure 2.8.** *Output of ins.html.*

## 2.3. COMMENT IN HTML CODE, COLOR, AND HYPERLINK

### 2.3.1. HTML Comments

HTML comments is a line that is not executed as a part of the program, only purpose is to be read by someone who is looking at the code is not displayed in the browser. Add comments to the following syntax:

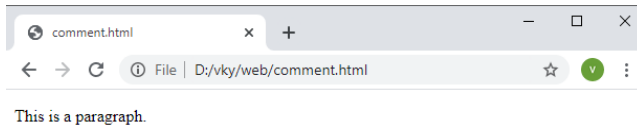
```
<!-- Write your comments here -->
```

Note: there is an exclamation point (!) in the start tag, but not use in the end tag.

Example: comment.html 2.9

```
<html>
<body>
<!-- This is a comment, and not displayed in the browser -->
<p>This is a paragraph.</p>
</body>
</html>
```

Output:



**Figure 2.9.** *Output of comment.html.*

### 2.3.2. HTML Colors

Colors are used to create the web page more attractive and beautiful. HTML colors are specific predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values.

**Color:** In HTML, a color can be specified by using a color name: red, yellow, blue, green etc.

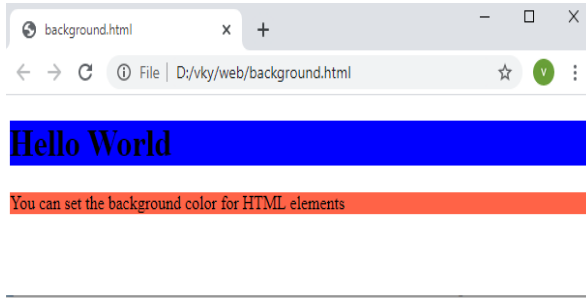
**Background color:** Use with background color HTML elements can set the background color of web page:

Example: background.html 2.10

```
<html>
<body>
<h1 style="background-color:Blue;">Hello World</h1>
<p style="background-color:Tomato;">You can set the background
color for HTML elements </p>
</body>
</html>
```

Output:





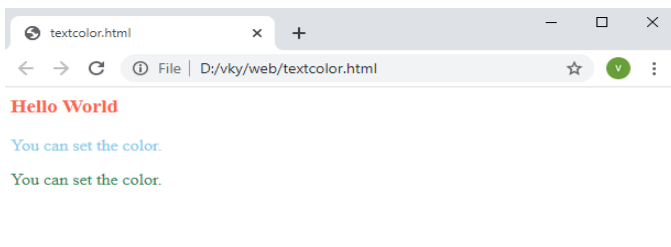
**Figure 2.10.** *Output of background.html.*

**Text color:** Use with color HTML elements can set the color of text in webpage.

Example: textcolor.html 2.11

```
<html>
<body>
<h3 style="color:Tomato;">Hello World</h3>
<p style="color:skyBlue;"> You can set the color.</p>
<p style="color:SeaGreen;"> You can set the color.</p>
</body>
</html>
```

Output:



**Figure 2.11.** *Output of textcolor.html.*

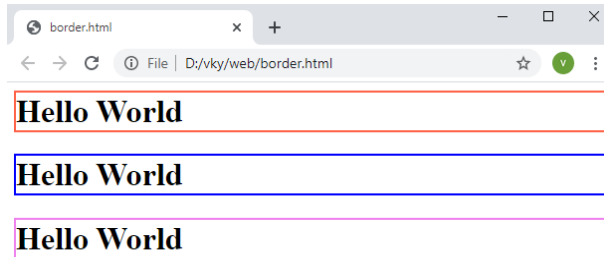
**Border color:** Use with border HTML elements can set the color of borders in your webpage.

Example: border.html 2.12

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

```
<h1 style="border: 2px solid Tomato;">Hello World</h1>
<h1 style="border: 2px solidBlue;">Hello World</h1>
<h1 style="border: 2px solid Violet;">Hello World</h1>
</body>
</html>
```

Output:



**Figure 2.12:** *Output of border.html.*

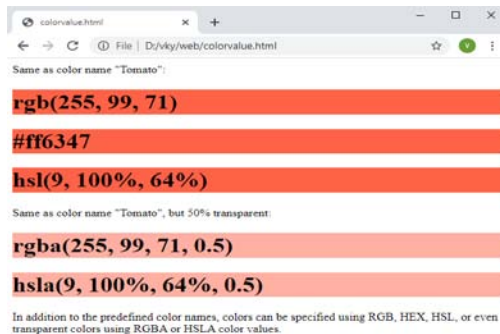
- **Color values:** In HTML, colors can also be represented using RGB values, HEX values, HSL values, RGBA values, and HSLA values.
- **Hexadecimal style:** In this style, we identify the color in 6 digit hexadecimal number (from 0 to F) and it is indicated by ‘#.’ In HEX values, first two digits indicate red color, next two green colors and the last two blue colors.
- **RGB style [Red Green Blue]:** In this style, we need to give 3 numbers indicated red, green and blue colors respectively required in the mixed color. The range of each color is from 0 to 255.
- **RGBA style [Red Green Blue Alpha]:** This style allows us to make the color transparent according and Alpha indicates the degree of transparency. The range of green, blue and red is from 0 to 255 and that of alpha is from 0 to 1.
- **HSL colors:** In this style, ‘H’ stands for hue, ‘S’ for Saturation and ‘L’ for Lightness, HSL color values.
- **Syntax:** hsl (hue, saturation, lightness)
- **Hue** is the color of the image itself. Its range is from 0 to 360. 0 is for red, 120 is for green and 240 is for blue.

- **Saturation** is the intensity/purity of the hue. 0% is for a shade of gray and 100% is the full color. When color is fully saturated, the color is considered in purest/truest version.
- **Lightness** is the color space's brightness. 0% is for black, 100% is for white.

Example: colorvalue.html 2.13

```
<html>
<body>
<p>Same as color name "Tomato":</p>
<h1 style="background-color:rgb(255, 99, 71);">rgb(255, 99, 71)</h1>
<h1 style="background-color:#ff6347;">#ff6347</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">hsl(9, 100%,
64%)</h1>
<p>Same as color name "Tomato," but 50% transparent:</p>
<h1 style="background-color:rgba(255, 99, 71, 0.5);">rgba(255, 99, 71,
0.5)</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">hsla(9, 100%,
64%, 0.5)</h1>
<p>In addition to the predefined color names, colors can be specified
using RGB, HEX, HSL, or even transparent colors using RGBA or HSLA
color values.</p>
</body>
</html>
```

Output:



**Figure 2.13:** *Output of colorvalue.html.*

### 2.3.3. Hyperlink

It is an association or link from one web page to another. A link has two ends, an anchor and path of resource. The link begins at the “source” anchor and points to the “destination” anchor, which may be any Web resource such as a text, an image, a video clip, a sound bite, a program, an HTML document. Links are specified in the “a” tag; <a> tag defines a hyperlink.

1. Link syntax:

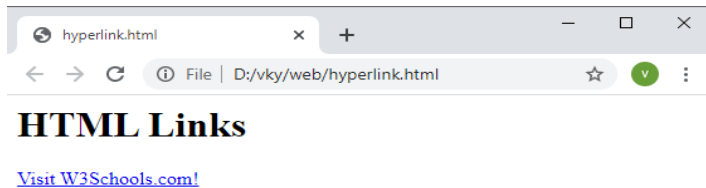
```
<a href=“url”>link text</a>
```

- The most important attribute of the <a> element is the href attribute, which indicates the link’s destination.
- The *link text* is the part that will be visible to the reader.
- Clicking on the link text, will send the reader to the specified URL address.

Example: hyperlink.html 2.14

```
<html>
<body>
<h1>HTML Links</h1>
<p><a href=“https://www.w3schools.com/”>Visit W3Schools.com!</a></p>
</body>
</html>
```

**Output:**



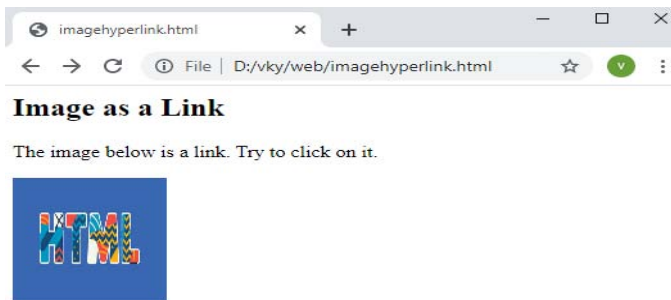
**Figure 2.14.** *Output of hyperlink.html.*

2. **Use an Image as a Link:** To use an image as a hyperlink, put the <img> tag inside the <a> tag.

Example: imagehyperlink.html 2.15

```
<html>
<body>
<h2>Image as a Link</h2>
<p>The image below is a link. Try to click on it.</p>
<a href="default.asp"></a>
</body>
</html>
```

Output:



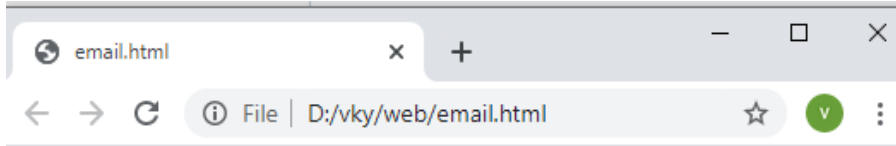
**Figure 2.15:** Output of *imagehyperlink.html*.

**3. Link to an Email Address:** Use `mailto:` inside the `href` attribute to create a link that opens the user's email.

Example: email.html 2.16

```
<html>
<body>
<h2>Link to an Email Address</h2>
<p>To create a link that opens in the user's email program (to let them
send a new email), use mailto: inside the href attribute:</p>
<p><a href="mailto:vinod. it210@gmail. com">Send email</a></p>
</body>
</html>
```

Output:



## Link to an Email Address

To create a link that opens in the user's email program (to let them send a new email), use `mailto:` inside the `href` attribute:

[Send email](#)

**Figure 2.16.** *Output of email.html.*

## 2.4. LISTS AND TABLES

### 2.4.1. List

A list is a record of small part of information, such as item's names, generally written or printed with a single item on each line and ordered in a way that makes a particular item easy to find.

**For example:**

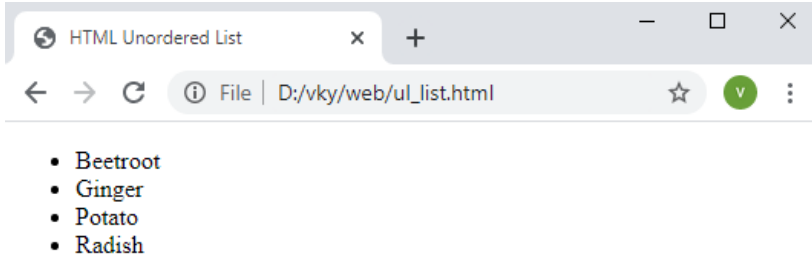
- A shopping list
- To-do list
- 1. **Lists in HTML:** HTML proposes three ways for identify lists of information. All lists have to contain one or more list elements. The three types of lists that can be used in HTML are:
  - **ul:** An unordered list, this will list items with plain bullets.
  - **ol:** An ordered list, this will use different format of numbers to list your items.
  - **dl:** A definition list, this arranges your items as the same way of a dictionary.

**The Unordered List:** An unordered list starts with the “`<ul>`” tag end with “`</ul>`,” each list of item define inside the `<ul>` tag starts with the “`<li>`” tag and end with “`</li>`.” These lists of items are marked with small black circles or bullets by default.

**Example: ul\_list.html 2.17**

```
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
<ul>
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ul>
</body>
</html>
```

Output:



**Figure 2.17.** *Output of ul\_list.html.*

- 2. The type Attribute:** You can use **type** attribute for `<ul>` tag to indicate the type of bullets. By default, it is a disc. Following are the feasible options for `<ul>` tag:

```
<ul type = "square">
```

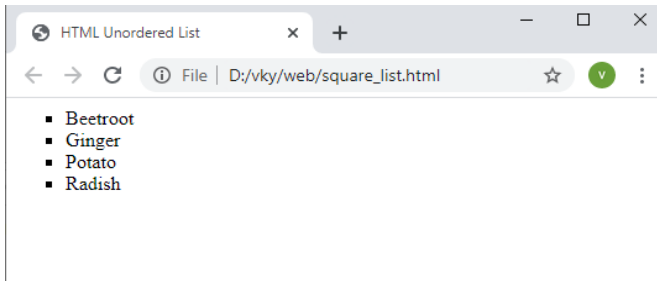
```
<ul type = "disc">
```

```
<ul type = "circle">
```

**Example: square\_list.html 2.18**

```
<html>
<head>
```

```
<title>HTML Unordered List</title>
</head>
<body>
<ul type = "square">
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ul>
</body>
</html>
```

**Output:**

**Figure 2.18:** *Output of square\_list.html.*

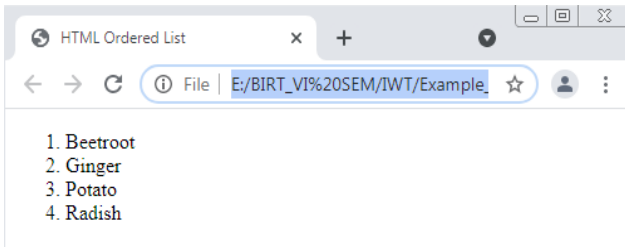
- 3. The ordered list:** An ordered list starts with the “<ol>” tag and ends with “</ol>.” Each list of item define inside the <ol> tag starts with the “<li>” tag and end with “</li>” tag. These lists of items are marked with numbers by default.

**Example: ol\_list.html 2.19**

```
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol>
```



```
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

**Output:**

**Figure 2.19.** *Output of ol\_list.html.*

- 4. The type Attribute:** You can use type attribute for `<ol>` tag to indicate the types of numbering, by default, it is a number. Following are the probable attribute options for `<ol>` tag:

**`<ol type = "1">` – Default-Case Numerals.**

**`<ol type = "I">` – Upper-Case Numerals.**

**`<ol type = "i">` – Lower-Case Numerals.**

**`<ol type = "A">` – Upper-Case Letters.**

**`<ol type = "a">` – Lower-Case Letters.**

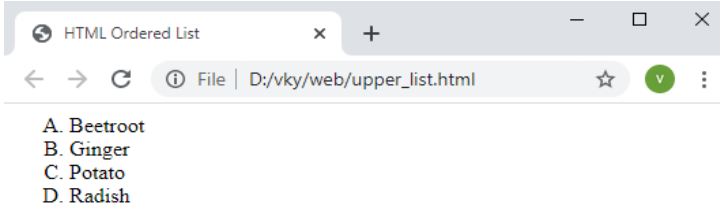
**Example: upper\_list.html 2.20**

```
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type = "A">
<li>Beetroot</li>
<li>Ginger</li>
```

```

</li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>

```

**Output:****Figure 2.20.** *Output of upper\_list.html.*

- 5. The description list:** A description list is a list of terms, through an explanation of each term. The description list defines with `<dl>` tag, the `<dt>` tag defines the term name, and the `<dd>` tag explain each term.

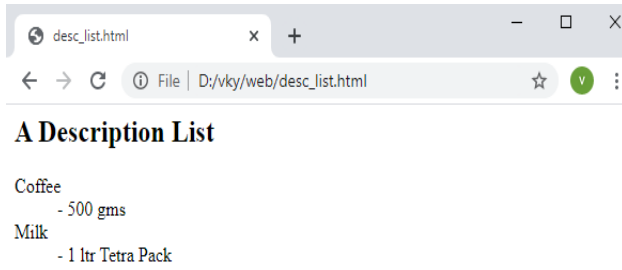
Example: desc\_list.html 2.21

```

<html>
<body>
<h2>A Description List</h2>
<dl>
<dt>Coffee</dt>
<dd>- 500 gms</dd>
<dt>Milk</dt>
<dd>- 1 ltr Tetra Pack</dd>
</dl>
</body>
</html>

```

Output:



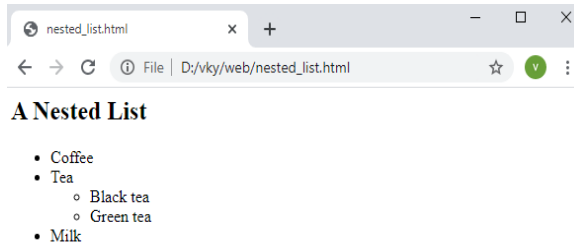
**Figure 2.21.** *Output of desc\_list.html.*

**6. Nested List:** A nested list is a list which has a list within a list.

Example: nested\_list.html 2.22

```
<html>
<body>
<h2>A Nested List</h2>
<ul>
<li>Coffee</li>
<li>Tea
  <ul>
    <li>Black tea</li>
    <li>Green tea</li>
  </ul>
</li>
<li>Milk</li>
</ul>
</body>
</html>
```

Output:



**Figure 2.22.** *Output of nested\_list.html.*

### 2.4.2. Tables

A table is a collection or representation of data in rows and columns form. Tables are broadly used in communication, research, and data analysis.

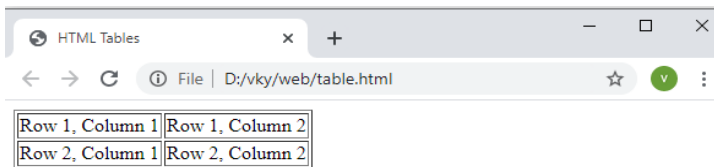
- Tables are useful for different tasks such as presenting text information and numerical data.
  - Tables can be used to evaluate two or more items in tabular form layout.
  - Tables are used to create databases.
1. **Define tables:** In HTML table is defined the starts with “<table>” tag and end with “</table>” tag. Each table inside row is defined with the “<tr>” tag and a table header is defined with the “<th>” tag. By default, a table heading are bold and centered, a table data/cell is defined with the “<td>” tag.

The **border** is an attribute of <table> tag and it is used to set a border across all the cells. If you do not need a border, then you can use border = “0.”

Example: table.html 2.23

```
<html>
<head>
<title>HTML Tables</title>
</head>
<body>
<table border = “1”>
<tr>
```

```
<td>Row 1, Column 1</td>
<td>Row 1, Column 2</td>
</tr>
<tr>
<td>Row 2, Column 1</td>
<td>Row 2, Column 2</td>
</tr>
</table>
</body>
</html>
```

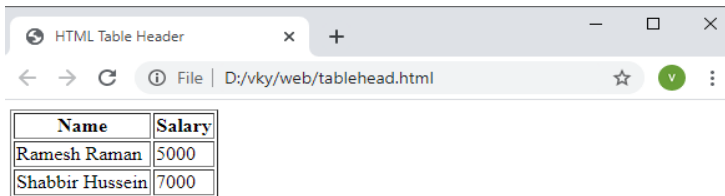
**Output:**

**Figure 2.23.** *Output of table.html.*

**Example: tablehead.html 2.24**

```
<html>
<head>
<title>HTML Table Header</title>
</head>
<body>
<table border = "1">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
```

```
<td>5000</td>
</tr>
<tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
</body>
</html>
```

**Output:**

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

**Figure 2.24.** *Output of tablehead.html.*

## 2.5. IMAGES AND FORMS

### 2.5.1. Images

1. **Insert images on a webpage:** The “<img>” tag is used to insert images on a webpage. The “<img>” tag is an empty tag, which means it can hold only a list of attributes and it has no closing tag.

**Syntax:** <img src=“url” alt=“some\_text”>

Attribute:

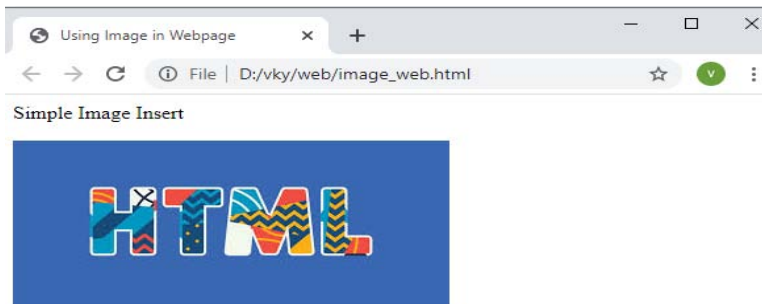
- **src:** source refer as to src. Each image tag has a src attribute which notify the browser where to find the image you want to display and URL of the image provided points to the location where the image is stored.

- **alt:** If the image cannot be displayed then the alt attribute acts as an alternative for the image. The value of the alt attribute is a user-defined text.

Example: image\_web.html 2.25

```
<html>
<head>
<title>Using Image in Webpage</title>
</head>
<body>
<p>Simple Image Insert</p>
<img src = "/html/images/test. png" alt = "Test Image" />
</body>
</html>
```

Output:



**Figure 2.25.** Output of image\_web.html.

2. **Set image location:** Generally we keep all the images in a separate directory, keep a HTML file test. htm in home directory and create a subdirectory images inside the home directory where we will keep our image test.png. Assume image location is "image/test.png" for HTML code.

```
<html>
<head>
<title>Using Image in Webpage</title>
</head>
```

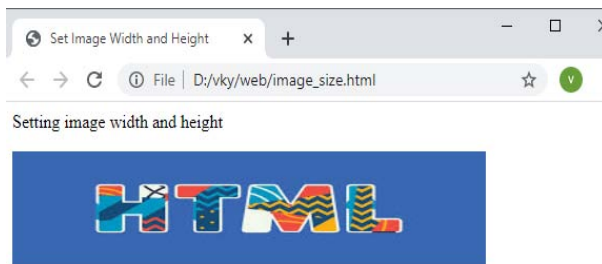
```
<body>
<p>Simple Image Insert</p>
<img src = "/html/images/test. png" alt = "Test Image" />
</body>
</html>
```

- 3. Set image width/height:** with the help of width and height attributes can set image width and height based on requirement and can set the image width and height in terms of either pixels or percentage of its actual size.

Example: image\_size.html 2.26

```
<html>
<head>
<title>Set Image Width and Height</title>
</head>
<body>
<p>Setting image width and height</p>
<img src = "/vky/web/images.png" alt = "Test Image" width = "450"
height = "100"/>
</body>
</html>
```

Output:



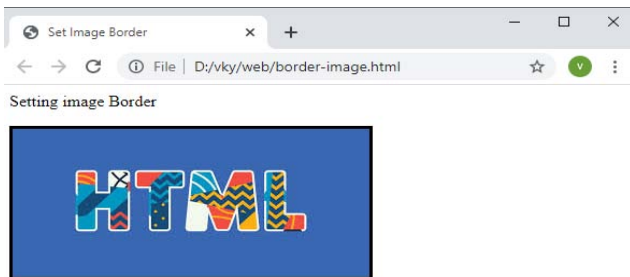
**Figure 2.26.** Output of *image\_size.html*.

- 4. Set image border:** Using border attribute, make border around image; you can set border thickness in terms of pixels. If thickness set 0 means no border around the image.



**Example: border\_image.html 2.27**

```
<html>
<head>
<title>Set Image Border</title>
</head>
<body>
<p>Setting image Border</p>
<img src = "/vky/web/images.png" alt = "Test Image" border = "3"/>
</body>
</html>
```

**Output:**

**Figure 2.27.** *Output of border\_image.html.*

- 5. Set image alignment:** Image will align at the left side of the page by default, but align attribute can use to set it in the center or right.

**Example: align.html 2.28**

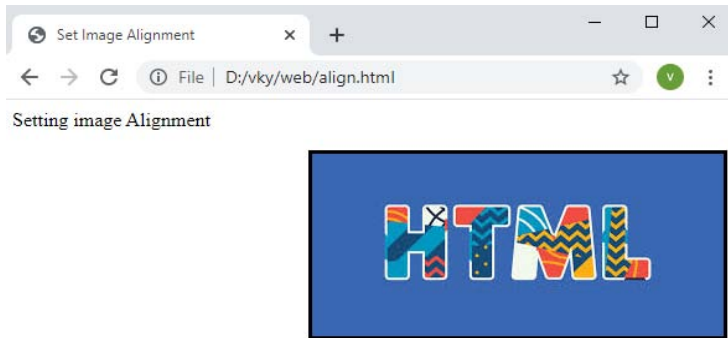
```
<html>
<head>
<title>Set Image Alignment</title>
</head>
<body>
<p>Setting image Alignment</p>
<img src = "/vky/web/images.png" alt = "Test Image" border = "3">
```

```
align = "right"/>
```

```
</body>
```

```
</html>
```

### Output:



**Figure 2.28.** *Output of align.html.*

## 2.5.2. Forms

HTML Forms are required to gather data from the website visitor and save in database. For example, when users want to fill registration forms, you would like to collect information such as name, email, addresses, credit card, etc. There are various elements of form available like text fields, text area fields, drop-down menus, radio buttons, checkboxes, etc.

Example: form.html

```
<html>
```

```
<head>
```

```
<title>Simple form</title>
```

```
</head>
```

```
<body>
```

```
<form>
```

```
User ID: <input type = "text" name = "user_id" />
```

```
<br>
```

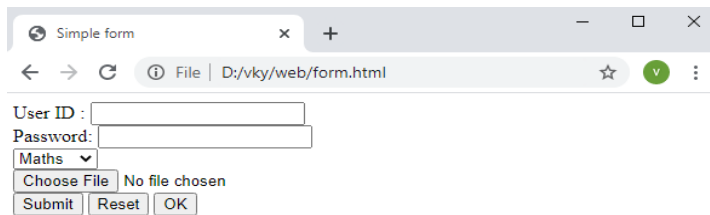
```
Password: <input type = "password" name = "password" /><br>
```

```

<select name = "dropdown">
<option value = "Maths" selected>Maths</option>
<option value = "Physics">Physics</option>
</select><br>
<input type = "file" name = "fileupload" accept = "image/*" /><br>
<input type = "submit" name = "submit" value = "Submit" />
<input type = "reset" name = "reset" value = "Reset" />
<input type = "button" name = "ok" value = "OK" />
</form>
</body>
</html>

```

Output:



**Figure 2.29.** *Output of form.html.*

1. **Form action attribute:** The action attribute describe where to send the form-data when a form is submitted. The HTML <form> tag is used to create an HTML form.

**Syntax:**

```

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

```

Example: form\_action.html 2.30

```

<html>
<body>
<h1>The form action attribute</h1>
<form action="/action_page. php">


```

```
<label for="fname">First name:</label>
<input type="text" id="fname" name="fname"><br><br>
<label for="lname">Last name:</label>
<input type="text" id="lname" name="lname"><br><br>
<input type="submit" value="Submit">
</form>
```

Click the “Submit” button and the form-data will be sent to a page on the

server called “action\_page.php.”

### Output:



**The form action attribute**

First name:

Last name:

Click the "Submit" button and the form-data will be sent to a page on the server called "action\_page.php".

**Figure 2.30.** *Output of form\_action.html.*

## CHAPTER 3

# INTRODUCTION TO CSS

## CONTENTS

3.1. Style Sheets: Need of CSS .....	62
3.2. Basic Structure of CSS.....	64
3.3. How to Use CSS in Webpage.....	69
3.4. CSS Properties for Background Images and Colors .....	71
3.5. CSS Properties for Manipulating Texts and Fonts .....	78
3.6. Borders and Margins.....	88
3.7. Padding and Lists .....	95
3.8. Positioning in Css.....	101
3.9. Overview and Features of CSS2 and CSS3 .....	103

Cascading style sheets, referred to as CSS, it is an easy web design language that process of making web pages presentable and interactive. CSS control the look and feel part of a web page. With CSS, you can manage the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, background images or colors are used, layout designs, and variations in display for different devices and screen sizes as well as a variety of other property. With the help of **CSS** control the style of web pages in an easy manner.

### 3.1. STYLE SHEETS: NEED OF CSS

CSS is a prerequisite for students and web professionals to become a great Software Engineer especially when they are working in Web Development field. Some of the advantages of learning CSS:

- **Create stunning web site:** CSS handles the look and feel part of a web page. You can control the color of the text, the style of fonts, the spacing between paragraphs, background images or colors are used, layout designs, and variations in display for different devices and screen sizes as well as a variety of other special effects.
- **Become a web designer:** If you want to start a career as a professional web designer, HTML and CSS designing is a must skill.
- **Control web:** CSS is easy to learn and it provides great control over the presentation of an HTML document. CSS is collective with the markup languages HTML or XHTML.
- **Learn other languages:** Once you aware of the basic of HTML and CSS then other correlated technologies like JavaScript, Php, or angular are become easier to understand and implement.

#### 3.1.1. Advantages of CSS

- **CSS saves time:** you can write CSS code once and then reuse same sheet in multiple HTML pages.
- **Pages load faster:** If you are using CSS, you do not need to write HTML tag attributes every time. Write single CSS rule of a tag and apply it to all the occurrences of that tag. So less code means quicker download times.

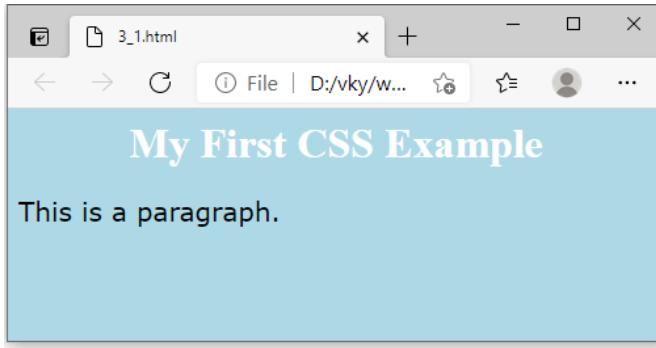
- **Easy maintenance:** To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML:** CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple device compatibility:** Style sheets allow content to be optimized for more than one type of device.
- **Global web standards:** Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's an excellent idea to start using CSS in all the HTML pages to make them compatible to future browsers.

Example of CSS: 3.1

```
<html>
<head>
<style>
body {
background-color: lightblue;
}
h1 {
color: white;
text-align: center;
}
p {
font-family: verdana;
font-size: 20px;
}
</style>
</head>
<body>
<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>
```

```
</body>
```

```
</html>
```

**Output:**

**Figure 3.1.** *Example of CSS: 3.1.*

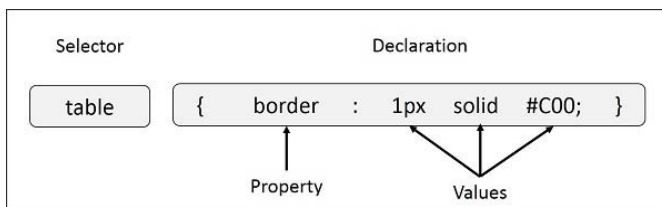
## 3.2. BASIC STRUCTURE OF CSS

CSS files consist of style rules that are interpreted by the web browser and then applied to the corresponding elements in your web document. A CSS style rule is prepared with three parts:

- **Selector:** A selector is an HTML tag at which a style will be applied. This could be any tag like `<h1>` or `<table>` etc.
- **Property:** A property is a type of attribute of HTML tag. Simply, all the HTML attributes are changed into CSS properties. They might be color, text-align, border etc.
- **Value:** Values are assigned to properties. Such as, color property can have value either *red* or *#F1F1F1* etc.

You be able to set CSS style rule syntax as Figure 3.2:

selector { property: value }



**Figure 3.2.** *CSS style rule.*



**Example:** you can identify a table border as follows:

```
table{ border: 2px solid #C00; }
```

At this time table is a **selector** and border is a **property** and given value 2px solid #C00 is the **value** of that property.

Once more example of **selector** to give a color to all level 1 headings:

```
h1 {  
color: #36CFFF;  
}
```

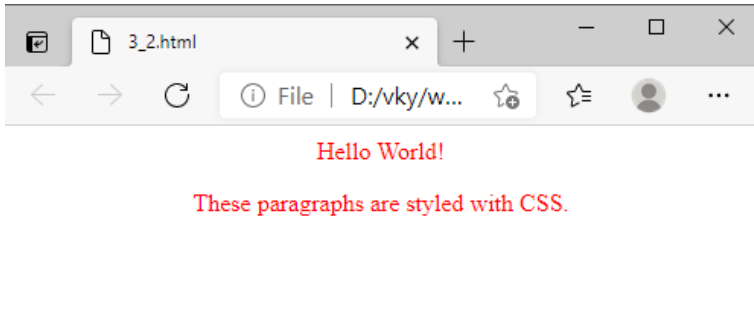
Example: 3.2

```
<html>  
<head>  
<style>  
p {  
color: red;  
text-align: center;  
}  
</style>  
</head>  
<body>  
<p>Hello World!</p>  
<p>These paragraphs are styled with CSS.</p>  
</body>  
</html>
```

Example Explained

- p is a **selector** in CSS (it points to the HTML element you want to style: <p>).
- color is a property, and red is the property value.
- text-align is a property, and center is the property value.

Output:



**Figure 3.3.** *Output of example 3.2.*

### 3.2.1. Types of Selectors

1. **The universal selectors:** for selecting elements of a specific type, the universal selector quite simply matches the name of any element type:

```
* {  
color: #000000;  
}
```

This rule renders the content of every element in our document in black.

Example: universal selector 3.3

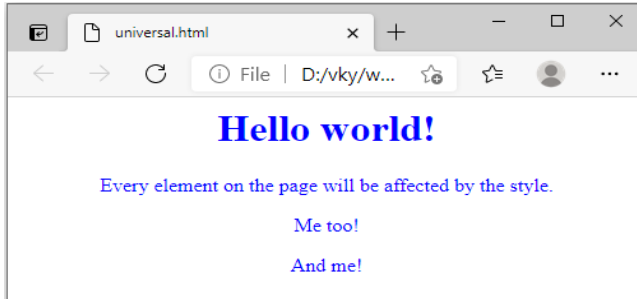
```
<html>  
<head>  
<style>  
* {  
text-align: center;  
color: blue;  
}  
</style>  
</head>  
<body>  
<h1>Hello world!</h1>  
<p>Every element on the page will be affected by the style.</p>  
<p id="para1">Me too!</p>
```

```
<p>And me!</p>
```

```
</body>
```

```
</html>
```

Output:



**Figure 3.4.** *Output of example 3.3.*

- 2. The CSS id selector:** The id selector uses the id attribute of an HTML element to select a particular element. The id of an element is unique within a page, to select an element with a specific id, write a hash (#) character, followed by the id of the element.

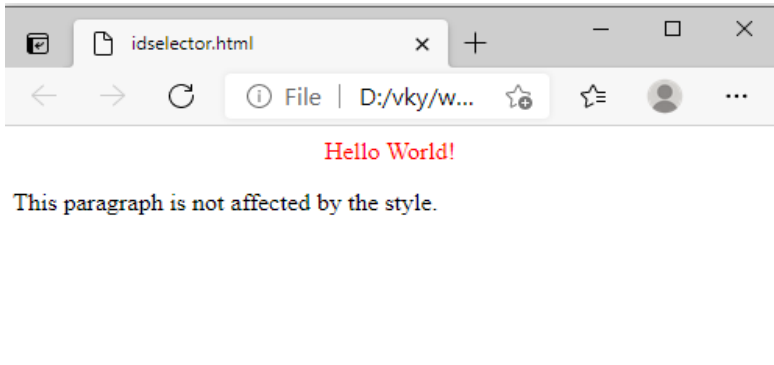
```
#para1 {  
text-align: center;  
color: red;  
}
```

Example: id selector 3.4

```
<html>  
<head>  
<style>  
#para1 {  
text-align: center;  
color: red;  
}  
</style>  
</head>  
<body>
```

```
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
</body>
</html>
```

Output:



**Figure 3.5.** *Output of example 3.4.*

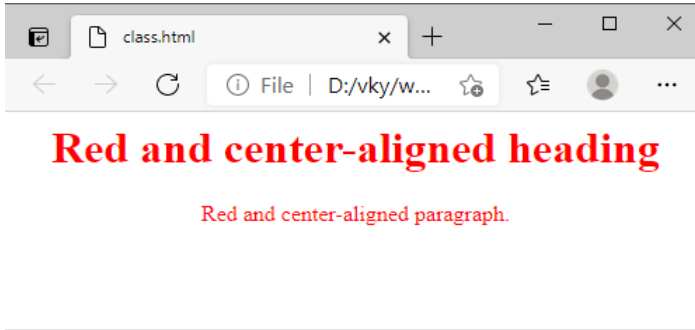
- 3. The class selectors:** The entire elements having that class will be formatted according to the defined rule.

```
.center {
color: #000000; }
```

#### **Example: class selector 3.5**

```
<html>
<head>
<style>
.center {
text-align: center;
color: red;
}
</style>
</head>
<body>
<h1 class="center">Red and center-aligned heading</h1>
```

```
<p class="center">Red and center-aligned paragraph.</p>
</body>
</html>
```

**Output:**

**Figure 3.6.** *Output of example 3.5.*

### 3.3. HOW TO USE CSS IN WEBPAGE

There are three methods for apply style sheet:

- External CSS
- Internal CSS
- Inline CSS

#### 3.3.1. External CSS

With the use an external style sheet, you can change the look of a whole website by changing just single file. All HTML page must contain a reference to the external style sheet file inside the `<link>` element, inside the head section. External styles are defined within the `<link>` element, inside the `<head>` section of an HTML page.

**Example: External CSS 3.6**

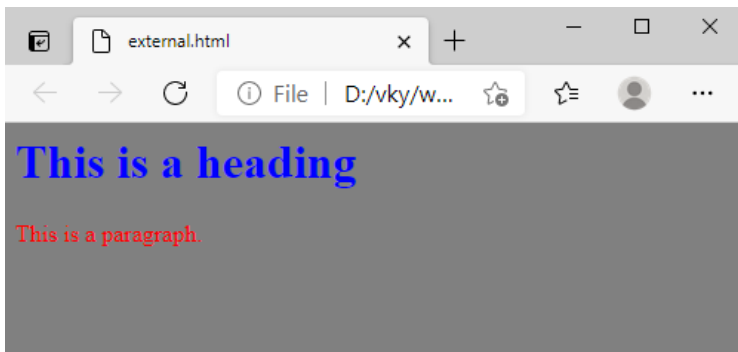
```
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>
```

```
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

An external style sheet can be written in any text editor, and must be saved with a CSS extension. The external CSS file should not contain any HTML tags. Here is how the “mystyle.css” file looks like: “mystyle.css”

```
body {
background-color: gray;
}
h1 {
color: blue;
}
p {
color: red;}
```

**Output:**



**Figure 3.7.** *Output of example 3.6.*

### 3.3.2. Internal CSS

An internal style sheet may be used if one single HTML page has a unique style. The internal style is defined inside the `<style>` element, inside the head section.

Example: Refer previous example 3.1, 3.2, 3.3, 3.4, 3.5 as Internal CSS

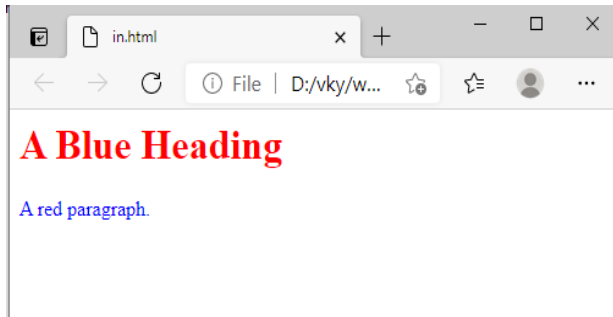
### 3.3.3. Inline CSS

An inline style may be applying a unique style for a single element. To use inline styles, add the style attribute to the applicable element. The style attribute can hold any CSS property. Inline styles are defined within the “style” attribute of the applicable element.

Example: inline CSS 3.7

```
<html>
<body>
<h1 style="color:red;">A Blue Heading</h1>
<p style="color:blue;">A red paragraph.</p>
</body>
</html>
```

**Output:**



**Figure 3.8.** *Output of example 3.7.*

## 3.4. CSS PROPERTIES FOR BACKGROUND IMAGES AND COLORS

The CSS background properties are used to describe the background effects form elements.

- The **background-color** property is used to set the background color of an element.
- The **background-image** property is used to set the background image of an element.

- The **background-repeat** property is used to manage the repetition of an image in the background.
- The **background-position** property is used to manage the position of an image in the background.
- The **background-attachment** property is used to control the scrolling of an image in the background.
- The **background** property is used as shorthand to specify a number of other background properties.

### 3.4.1. CSS Background-Color

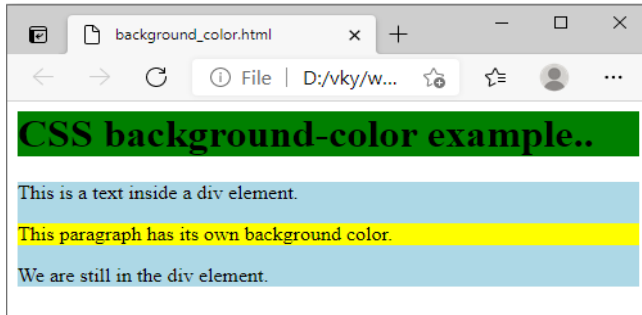
The background-color property identifies the background color of an element. Here, the <h1>, <p>, and <div> elements will have different background colors:

#### Example: background\_color 3.8

```
<html>
<head>
<style>
h1 {
background-color: green;
}
div {
background-color: lightblue;
}
p {
background-color: yellow;
}
</style>
</head>
<body>
<h1>CSS background-color example!</h1>
<div> This is a text inside a div element.
<p>This paragraph has its own background color.</p>
We are still in the div element.
```



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

**Output:**

**Figure 3.9.** *Output of example 3.8.*

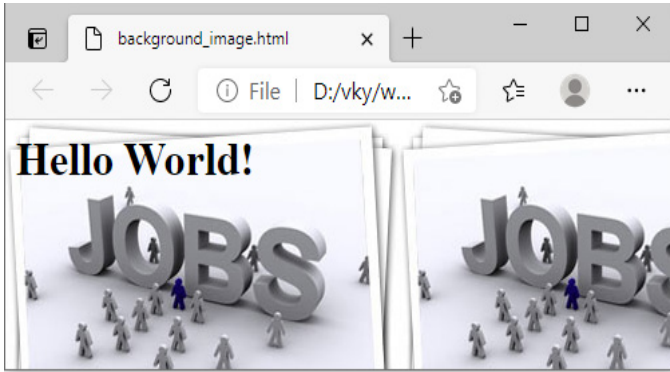
### 3.4.2. CSS Background-Image

The background-image property identifies an image use as the background of an element.

Example: background-images 3.9

```
<html>
<head>
<style>
body {
background-image: url("job.jpg");
background-color: #cccccc;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
</body>
</html>
```

Output:



**Figure 3.10.** *Output of example 3.9.*

### 3.4.3. Repeat the Background Image

The next example demonstrates how to repeat the background image if an image is small. You can use no-repeat value for background-repeat property if you don't want to repeat an image, in this case image will display only once.

#### **Example: repeat\_image 3.10**

```
<html>
<head>
<style>
body {
background-image: url("job.jpg");
background-repeat: repeat;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
</body>
</html>
```

**Output:**

**Figure 3.11.** *Output of example 3.10.*

- to repeat the background image vertically use:  
background-repeat: repeat-y;
- to repeat the background image horizontally use:  
background-repeat: repeat-x;

### 3.4.4. CSS Background-Position

The background-position property is used to specify the position of the background image.

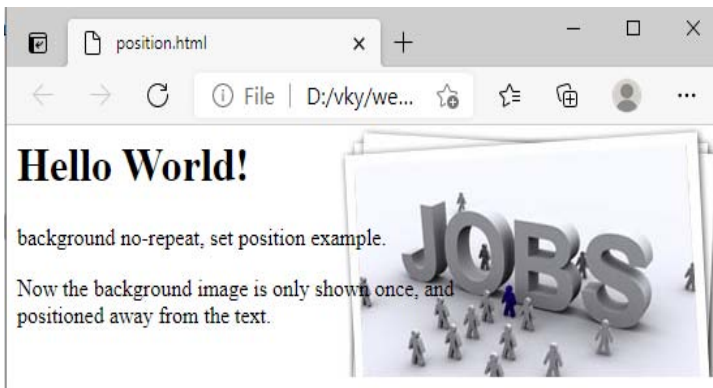
Position the background image in the top-right corner:

```
body {  
  background-image: url("job.jpg");  
  background-repeat: no-repeat;  
  background-position: right top;  
}
```

Example: Position\_3.11

```
<html>  
<head>  
<style>  
body {  
  background-image: url("job.jpg");  
  background-repeat: no-repeat;  
  background-position: right top;  
}
```

```
margin-right: 200px;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<p> background no-repeat, set position example.</p>
<p>Now the background image is only shown once, and positioned
away from the text.</p>
</body>
</html>
```

**Output:**

**Figure 3.12.** *Output of example 3.11.*

### 3.4.5. CSS Background – Shorthand Property

To shorten the code, it is also possible to identify all the background properties in one single property. This is known as shorthand property. In its place of writing:

```
body {
background-color: #ffffff;
background-image: url("img_tree.png");
background-repeat: no-repeat;
```

```
background-position: right top;
}
```

You can employ the shorthand property `background`: Use the shorthand property to set the background properties in single declaration:

```
body {
background: #ffffff url("img_tree.png") no-repeat right top;
}
```

Example: shorthand\_3.12

```
<html>
<head>
<style>
body {
background: #ffffff url("job.jpg") no-repeat right top;
margin-right: 200px;
}
</style>
</head>
<body>
<h1>The background Property</h1>
```

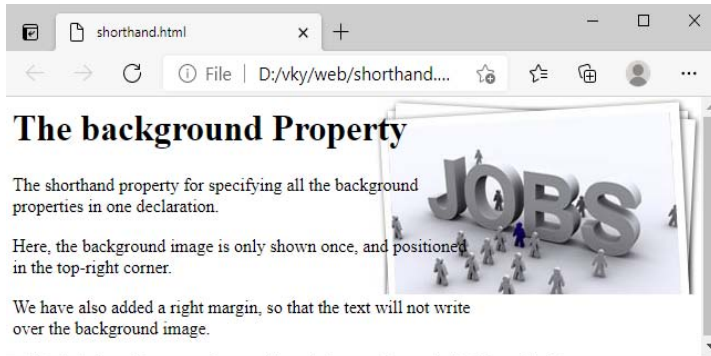
<p>The shorthand property for specifying all the background properties in one declaration.</p>

<p>Here, the background image is only shown once, and positioned in the top-right corner.</p>

<p>We have also added a right margin, so that the text will not write over the background image.</p>

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

Output:



**Figure 3.13.** *Output of example 3.12.*

## 3.5. CSS PROPERTIES FOR MANIPULATING TEXTS AND FONTS

### 3.5.1. Manipulate Text

Using CSS properties you can put following text properties of an element:

- The **color** property is used to set the color of a text.
- The **direction** property is used to place the text direction.
- The **letter-spacing** property is used to add or subtract space between the letters that make up a word.
- The **word-spacing** property is used to add or subtract space between the words of a sentence.
- The **text-indent** property is used to indent the text of a paragraph.
- The **text-align** property is used to align the text of a document.
- The **text-decoration** property is used to underline, over line, and strikethrough text.
- The **text-transform** property is used to capitalize text or convert text to uppercase or lowercase letters.
- The **text-shadow** property is used to set the text shadow around a text.

#### 3.5.1.1. Text Color

The color property is used to put the color of the text. The color is specified by:

- a color name – like “red”
- a HEX value – like “#ff0000”
- an RGB value – like “rgb(255,0,0)”

Example: textcolor\_3.13

```
<html>
```

```
<head>
```

```
<style>
```

```
body {
```

```
color: blue;
```

```
}
```

```
h1 {
```

```
color: green;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>This is heading line</h1>
```

<p>This is an ordinary paragraph. Notice that this text is blue. The default text color for a page is defined in the body selector.</p>

```
<p>Another paragraph.</p>
```

```
</body>
```

```
</html>
```

### Output:



**Figure 3.14.** *Output of example 3.13.*

### 3.5.1.2. *Text Color and Background Color*

In this example, we define both the background-color property and the color property:

Example: text\_background\_3.14

```
<html>
```

```
<head>
```

```
<style>
```

```
body { background-color: lightgray;
```

```
color: blue;
```

```
}
```

```
h1 {
```

```
color: green;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>This is heading line</h1>
```

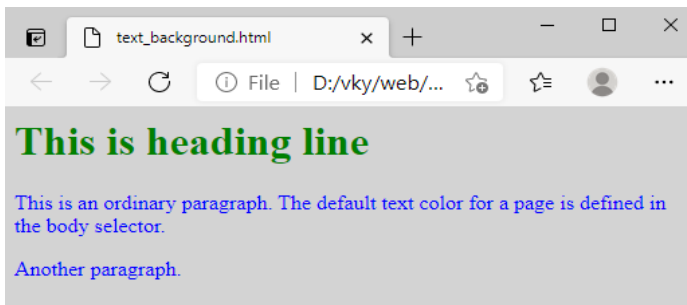
```
<p>This is an ordinary paragraph. The default text color for a page is  
defined in the body selector.</p>
```

```
<p>Another paragraph.</p>
```

```
</body>
```

```
</html>
```

Output:



**Figure 3.15.** *Output of example 3.14.*



### 3.5.1.3. Text Direction

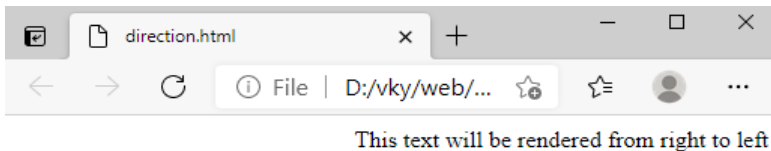
The direction properties can be used to change the text direction of an element. Possible values are *ltr* or *rtl*.

```
p {  
  direction: rtl;  
}
```

Example: direction\_3.15

```
<html>  
<head>  
</head>  
<body>  
<p style = "direction:rtl;">  
  This text will be rendered from right to left  
</p>  
</body>  
</html>
```

Output:



**Figure 3.16.** Output of example 3.15.

### 3.5.1.4. Space between Characters and Space between Words

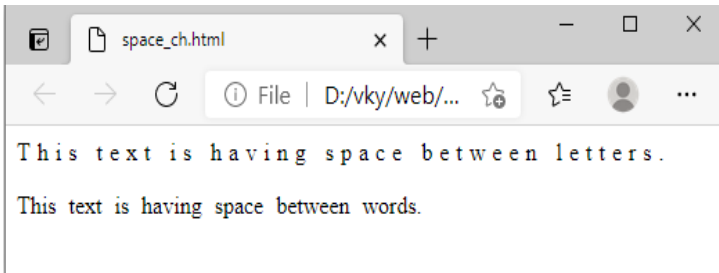
Set the space between characters. Possible values are normal or a number specifying space. Set the space between words. Possible values are normal or a number specifying space.

Example: space\_ch\_word\_3.16

```
<html>
```

```
<head>
</head>
<body>
  <p style = "letter-spacing:5px;"> This text is having space between
  letters.</p>
  <p style = "word-spacing:5px;">This text is having space between
  words.</p>
</body>
</html>
```

Output:



**Figure 3.17.** *Output of example 3.16.*

### ***3.5.1.5. Text Indent and Text Alignment***

The following example demonstrates how to indent the first line of a paragraph. A possible value is % or a number specifying indent space and demonstrates how to align a text. Possible values are left, right, center, justify.

Example: indent&align\_3.17

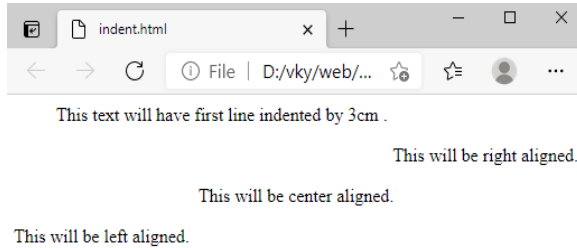
```
<html>
<head>
</head>
<body>
  <p style = "text-indent:3cm;">This text will have first line indented by
  1cm.</p>
  <p style = "text-align:right;">This will be right aligned.</p>
  <p style = "text-align:center;">This will be center aligned.</p>
```

```
<p style = "text-align:left;"> This will be left aligned.</p>
```

```
</body>
```

```
</html>
```

Output:



**Figure 3.18.** *Output of example 3.17.*

### 3.5.1.6. *Decorating the Text*

The following example demonstrates how to decorate a text. Possible values are none, underline, over line, line-through, and blink.

Example: decorating\_3.18

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<p style = "text-decoration:underline;">This will be underlined</p>
```

```
<p style = "text-decoration:line-through;">This will be striked through.
```

```
</p>
```

```
<p style = "text-decoration:overline;">This will have a over line.</p>
```

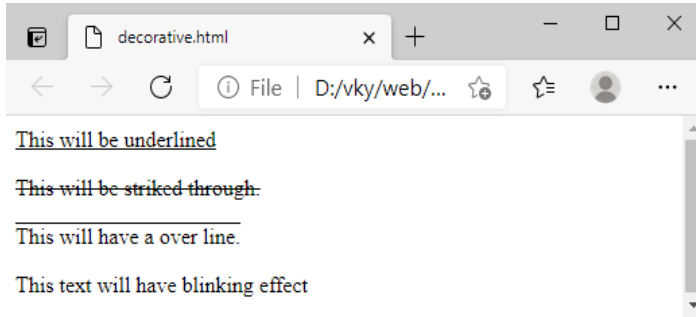
```
<p style = "text-decoration:blink;">This text will have blinking effect
```

```
</p>
```

```
</body>
```

```
</html>
```

Output:



**Figure 3.19.** *Output of example 3.18.*

### 3.5.1.7. Text Cases and Text Shadow

The following example demonstrates how to set the cases for a text. A possible value is none, capitalizes, uppercase, lowercase and demonstrates how to set the shadow around a text. This may not be supported by all the browsers.

Example: text\_3.19

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<p style = "text-transform:capitalize;"> This will be capitalized</p>
```

```
<p style = "text-transform:uppercase;"> This will be in uppercase</p>
```

```
<p style = "text-transform:lowercase;"> This will be in lowercase</p>
```

```
<p style = "text-shadow:4px 4px 8px blue;"> If your browser supports  
the CSS text-shadow property, this text will have a blue shadow.</p>
```

```
</body>
```

```
</html>
```

**Output:**



**Figure 3.20.** *Output of Example 3.19.*

### 3.5.2. CSS Font

The CSS font properties describe the font boldness, size, and the style of a text.

We can put following font properties of an element:

- The **font-style** property is used to make a font italic or oblique.
- The **font-variant** property is used to create a small-caps effect.
- The **font-weight** property is used to increase or decrease how bold or light a font appears.
- The **font-size** property is used to increase or decrease the size of a font.
- The **font** property is used as shorthand to specify a number of other font properties.

#### 3.5.2.1. The Font Style and Font Variant

Set the font style of an element. Possible values are normal, italic and oblique. Place the font variant of an element. Possible values are normal and small-caps.

##### **Example: fontstyle\_3.20**

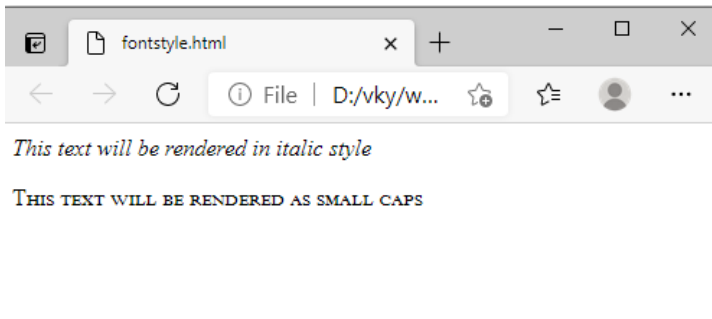
```
<html>
<head>
</head>
<body>
```

```

<p style = “font-style:italic;”>This text will be rendered in italic style</p>
<p style = “font-variant:small-caps;”>This text will be rendered as small caps</p>
</body>
</html>

```

### Output:



**Figure 3.21.** Output of example 3.20.

### 3.5.2.2. The Font Weight and Font Size

Put the font weight of an element, font-weight property provides the functionality how bold a font is. Possible values could be normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, and 900. Set the font size of an element. The font-size property is used to control the size of fonts. Possible values could be xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, size in pixels or in percentage.

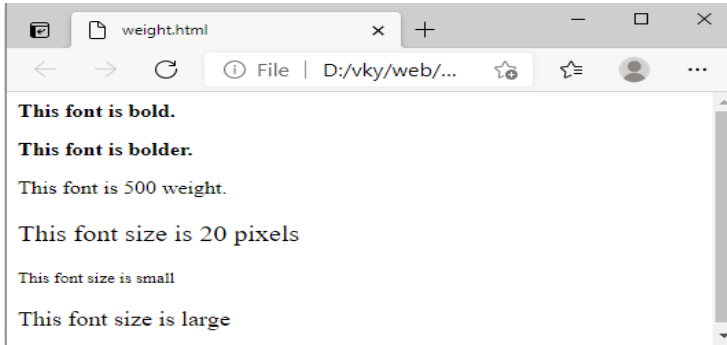
#### Example: weight\_size\_3.21

```

<html>
<head>
</head>
<body>
<p style = “font-weight:bold;”>This font is bold. </p>
<p style = “font-weight:bolder;”>This font is bolder. </p>
<p style = “font-weight:500;”>This font is 500 weight. </p>
<p style = “font-size:20px;”>This font size is 20 pixels</p>

```

```
<p style = "font-size:small;">This font size is small</p>
<p style = "font-size:large;">This font size is large</p>
</body>
</html>
```

**Output:**

**Figure 3.22.** *Output of example 3.21.*

### 3.5.2.3. Shorthand Property

You can use the font property to place all the font properties at once. For example –

Example: shorthand\_3.22

```
<html>
```

```
<head>
```

```
</head>
```

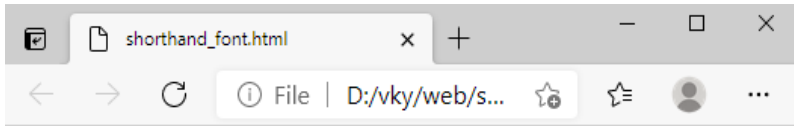
```
<body>
```

```
<p style = "font:italic small-caps bold 15px georgia;">Applying all the
properties on the text at once.</p>
```

```
</body>
```

```
</html>
```

Output:



***APPLYING ALL THE PROPERTIES ON THE TEXT AT ONCE.***

---

**Figure 3.23.** *Output of example 3.22.*

## 3.6. BORDERS AND MARGINS

The border properties allow you to specify how the border of the box representing an element should look. There are three properties of a border you can change –

- The border-color specifies the color of a border.
- The border-style specifies whether a border should be solid, dashed line, double line, or one of the other possible values.
- The border-width specifies the width of a border.

### 3.6.1. The Border-Color Property

The border-color property allows you to change the color of the border surrounding an element. You can individually change the color of the bottom, left, top and right sides of an element's border using the properties –

- Border-bottom-color changes the color of bottom border.
- border-top-color changes the color of top border.
- border-left-color changes the color of left border.
- border-right-color changes the color of right border.

Example shows the effect of all these properties –

Example: border-color-3.23

```
<html>
```

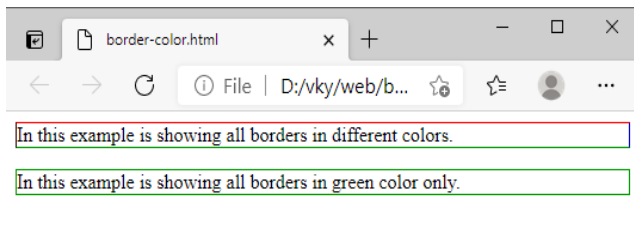
```
<head>
```

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

```
p.example1 {
```



```
border: 1px solid;
border-bottom-color: #009900; /* Green */
border-top-color: #FF0000; /* Red */
border-left-color: #330000; /* Black */
border-right-color: #0000CC; /* Blue */
}
p.example2 {
border: 1px solid;
border-color: #009900; /* Green */
}
</style>
</head>
<body>
<p class = "example1"> In this example is showing all borders in different
colors.</p>
<p class = "example2">In this example is showing all borders in green
color only. </p>
</body>
</html>
```

**Output:**

**Figure 3.24.** *Output of Example 3.23.*

### 3.6.2. CSS Border Style

The border-style property specifies what kind of border to display. The following values are allowed:

- dotted – Defines a dotted border
- dashed – Defines a dashed border
- solid – Defines a solid border
- double – Defines a double border

- groove – Defines a 3D grooved border. The effect depends on the border-color value
- ridge – Defines a 3D ridged border. The effect depends on the border-color value
- inset – Defines a 3D inset border. The effect depends on the border-color value
- outset – Defines a 3D outset border. The effect depends on the border-color value
- none – Defines no border
- hidden – Defines a hidden border

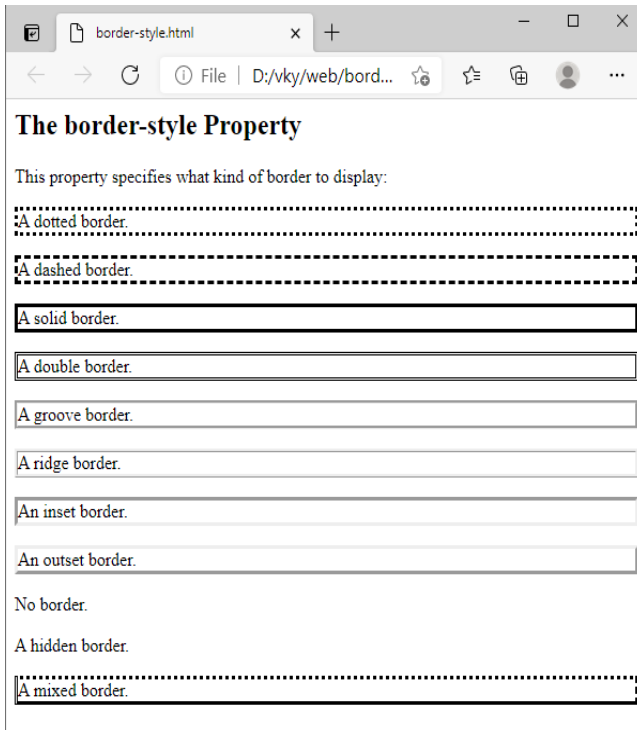
The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example: border-style-3.24

```
<html>
<head>
<style>
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
</style>
</head>
<body>
<h2>The border-style Property</h2>
<p>This property specifies what kind of border to display:</p>
<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
```

```
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
<p class="none">No border.</p>
<p class="hidden">A hidden border.</p>
<p class="mix">A mixed border.</p>
</body>
</html>
```

## Output:



**Figure 3.25.** *Output of example 3.24.*

### 3.6.3. The Border-Width Property

The border-width property allows you to set the width of an element borders. You can change the width of the bottom, top, left, and right borders of an element using properties –

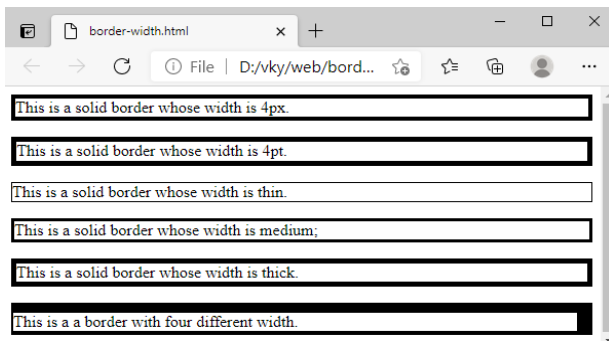
- **border-bottom-width** changes the width of bottom border.
- **border-top-width** changes the width of top border.

- **border-left-width** changes the width of left border.
- **border-right-width** changes the width of right border.

Example: border-width-3.25

```
<html>
<body>
  <p style = "border-width: 4px; border-style: solid;">This is a solid
border whose width is 4px.</p>
  <p style = "border-width:4pt; border-style: solid;"> This is a solid border
whose width is 4pt.</p>
  <p style = "border-width:thin; border-style: solid;">This is a solid
border whose width is thin.
</p>
  <p style = "border-width:medium; border-style: solid;">This is a solid
border whose width is medium;</p>
  <p style = "border-width:thick; border-style:s olid;">This is a solid
border whose width is thick.
</p>
  <p style = "border-bottom-width:4px;border-top-width:10px;border-
left-width: 2px;border-right-width:15px;border-style:solid;"> This is a
border with four different width.</p>
</body>
</html>
```

Output:



**Figure 3.26.** Output of example 3.25.

### 3.6.4. CSS Margins

CSS margins are used to create space around the element. We can set the different size of margins for individual sides (top, right, bottom, left). Margin properties can have following values:

1. Length in cm, px, pt, etc.
2. Width % of the element.
3. Margin calculated by the browser: auto.

Note: Negative values are allowed.

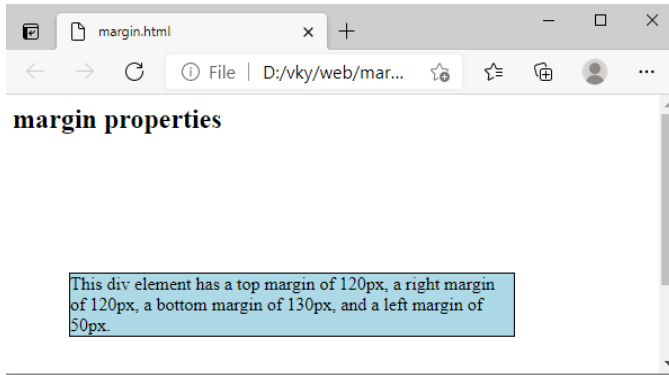
**Syntax:**

```
body
{
margin: size;
}
```

**Example: margin-3.26**

```
<html>
<head>
<style>
div {
    border: 1px solid black;
    margin-top: 120px;
    margin-bottom: 130px;
    margin-right: 120px;
    margin-left: 50px;
    background-color: lightblue;
}
</style>
</head>
<body>
<h2> margin properties</h2>
<div>This div element has a top margin of 120px, a right margin of
120px, a bottom margin of 130px, and a left margin of 50px.</div>
</body>
</html>
```

**Output:**



**Figure 3.27.** *Output of example 3.26.*

We have the following properties to set an element margin.

- The margin specifies a shorthand property for setting the margin properties in one declaration.
- The margin-bottom specifies the bottom margin of an element.
- The margin-top specifies the top margin of an element.
- The margin-left specifies the left margin of an element.
- The margin-right specifies the right margin of an element.

Example: margin-property-3.27

```
<html>
```

```
<head>
```

```
<style>
```

```
div {
```

```
    border: 1px solid black;
```

```
    margin: 25px 50px 75px 100px;
```

```
    background-color: lightblue;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h2>The margin property – 4 values</h2>
```

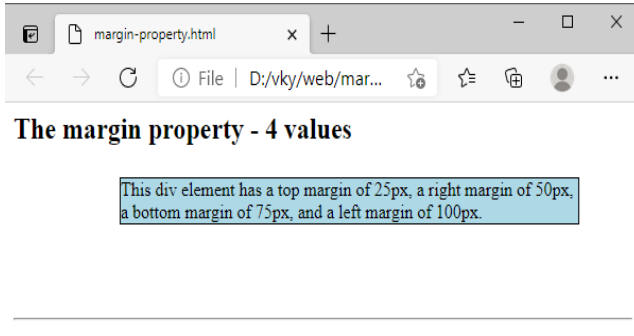
```
<div>This div element has a top margin of 25px, a right margin of 50px,  
a bottom margin of 75px, and a left margin of 100px.</div>
```

```
<hr>
```

```
</body>
```

```
</html>
```

**Output:**



**Figure 3.28.** *Output of example 3.27.*

## 3.7. PADDING AND LISTS

### 3.7.1. Padding

The *padding* property allows you to specify how much space should appear between the content of an element and its border. The value of this attribute should be either a length, a percentage, or the word *inherits*. If the value is inherited, it will have the same padding as its parent element. If a percentage is used, the percentage is of the containing box. The following CSS properties can be used to control lists. You can also set different values for the padding on each side of the box using the following properties:

- The **padding-bottom** specifies the bottom padding of an element.
- The **padding-top** specifies the top padding of an element.
- The **padding-left** specifies the left padding of an element.
- The **padding-right** specifies the right padding of an element.

#### 3.7.1.1. *Padding-Bottom Property*

The *padding-bottom* property sets the bottom padding (space) of an element. This can take a value in terms of length of %.

Example: `padding-bottom: 3.28`

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

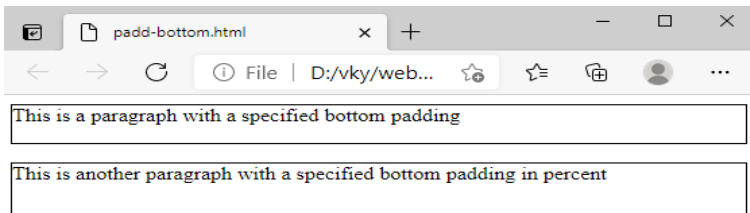
```
<p style = "padding-bottom: 15px; border:1px solid black;"> This is a  
paragraph with a specified bottom padding </p>
```

```
<p style = "padding-bottom: 5%; border:1px solid black;">This is  
another paragraph with a specified bottom padding in percent</p>
```

```
</body>
```

```
</html>
```

Output:



**Figure 3.29.** Output of example 3.28.

### 3.7.1.2. *Padding-Top Property*

The *padding-top* property sets the top padding (space) of an element. This can take a value in terms of length or %.

Example: padd-top-3.29

```
<html>
```

```
<body>
```

```
<p style = "padding-top: 15px; border:1px solid black;"> This is a  
paragraph with a specified top padding </p>
```

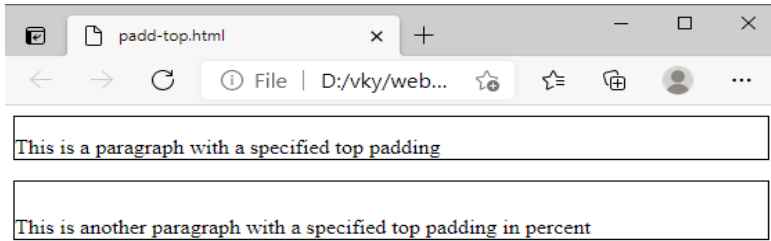
```
<p style = "padding-top: 5%; border:1px solid black;"> This is another  
paragraph with a specified top padding in percent</p>
```

```
</body>
```

```
</html>
```

Output:





**Figure 3.30.** *Output of example 3.29.*

### 3.7.1.3. *Padding-Left Property*

The *padding-left* property sets the left padding (space) of an element. This can take a value in terms of length or %.

Example: padd-left-3.30

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

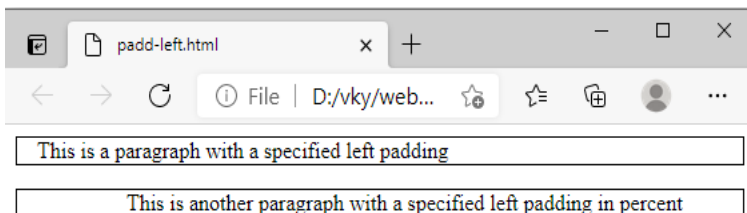
```
<p style = "padding-left: 15px; border:1px solid black;">This is a  
paragraph with a specified left padding</p>
```

```
<p style = "padding-left: 15%; border:1px solid black;">This is another  
paragraph with a specified left padding in percent</p>
```

```
</body>
```

```
</html>
```

**Output:**



**Figure 3.31.** *Output of example 3.30.*

#### 3.7.1.4. *Padding-Right Property*

The *padding-right* property sets the right padding (space) of an element. This can take a value in terms of length or %.

Example: padd-right-3.31

```
<html>
```

```
<body>
```

```
<p style = "padding-right: 15px; border: 1px solid black;">
```

```
This is a paragraph with a specified right padding </p>
```

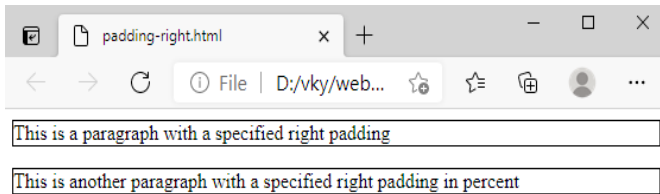
```
<p style = "padding-right: 5%; border: 1px solid black;">
```

```
This is another paragraph with a specified right padding in percent </p>
```

```
</body>
```

```
</html>
```

Output:



**Figure 3.32.** *Output of example 3.31.*

#### 3.7.2. *CSS List Properties*

In HTML, there are two main types of lists:

- unordered lists (<ul>) – the list items are marked with bullets.
- ordered lists (<ol>) – the list items are marked with numbers or letters.

The CSS list properties allow you to:

- Set different list item markers for ordered lists.
- Set different list item markers for unordered lists.
- Set an image as the list item marker.
- Add background colors to lists and list items.

### 3.7.2.1. *Different List Item Markers*

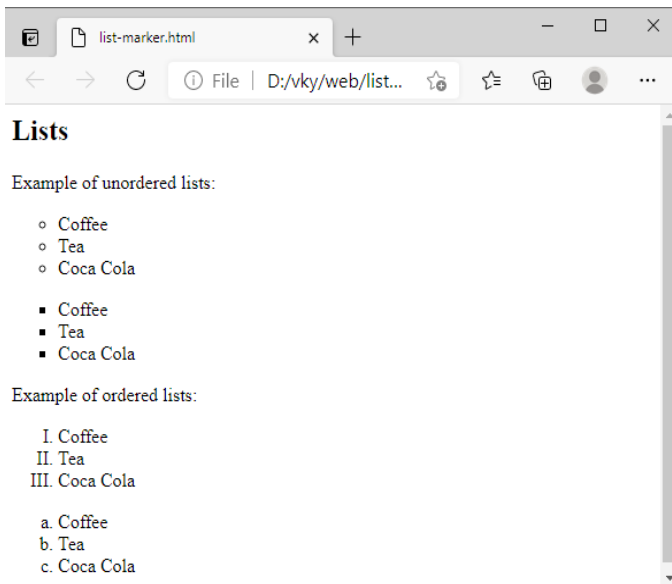
The list-style-type property specifies the type of list item marker.

Example: list-markert-3.32

```
<html>
<head>
<style>
ul.a {
list-style-type: circle;
}
ul.b {
list-style-type: square;
}
ol.c {
list-style-type: upper-roman;
}
ol.d {
list-style-type: lower-alpha;
}
</style>
</head>
<body>
<h2>Lists</h2>
<p>Example of unordered lists:</p>
<ul class="a">
<li>Coffee</li>
<li>Tea</li>
<li>Coca Cola</li>
</ul>
<ul class="b">
<li>Coffee</li>
<li>Tea</li>
```

```
<li>Coca Cola</li>
</ul>
<p>Example of ordered lists:</p>
<ol class="c">
<li>Coffee</li>
<li>Tea</li>
<li>Coca Cola</li>
</ol>
<ol class="d">
<li>Coffee</li>
<li>Tea</li>
<li>Coca Cola</li>
</ol>
</body>
</html>
```

Output:



**Figure 3.33.** *Output of example 3.32.*

## 3.8. POSITIONING IN CSS

CSS helps you to position your HTML element. You can put any HTML element at whatever location you like. You can specify whether you want the element positioned relative to its natural position in the page or absolute based on its parent element.

### 3.8.1. *Relative Positioning*

Relative positioning changes the position of the HTML element relative to where it normally appears. So “left:20” adds 20 pixels to the element’s LEFT position. You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

- Move left – Use a negative value for *left*.
- Move right – Use a positive value for *left*.
- Move up – Use a negative value for *top*.
- Move down – Use a positive value for *top*.

Example: relative-positioning 3.33

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<div style = “position:relative; left:80px; top:2px; background-color:yellow;”>
```

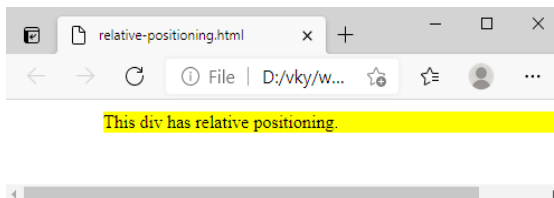
This div has relative positioning.

```
</div>
```

```
</body>
```

```
</html>
```

Output:



**Figure 3.34.** *Output of example 3.33.*

### 3.8.2. Absolute Positioning

An element with **position: absolute** is positioned at the specified coordinates relative to your screen top-left corner. You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

- Move left – Use a negative value for *left*.
- Move right – Use a positive value for *left*.
- Move up – Use a negative value for *top*.
- Move down – Use a positive value for *top*.

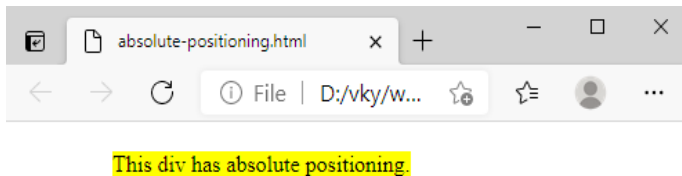
Example: absolute-positioning 3.34

```
<html>
<head>
</head>
<body>
  <div style = "position:absolute; left:80px; top:20px; background-
color:yellow;">
```

    This div has absolute positioning.

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

Output:



---

**Figure 3.35.** Output of example 3.34.

### 3.8.3. Fixed Positioning

Fixed positioning allows you to fix the position of an element to a particular spot on the page, regardless of scrolling. You can use two values *top* and *left*

along with the *position* property to move an HTML element anywhere in the HTML document.

- Move left – Use a negative value for *left*.
- Move right – Use a positive value for *left*.
- Move up – Use a negative value for *top*.
- Move Down – Use a positive value for *top*.

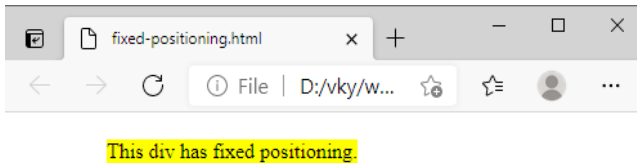
Example: fixed-positioning 3.35

```
<html>
<head>
</head>
<body>
  <div style = "position:fixed; left:80px; top:20px; background-
color:yellow;">
```

This div has fixed positioning.

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

Output:



**Figure 3.36.** Output of example 3.35.

## 3.9. OVERVIEW AND FEATURES OF CSS2 AND CSS3

### 3.9.1. CSS2 Overview

Cascading style sheets level 2 (CSS2) is the next version of cascading style sheets that developed by W3C. It's a declarative language used to enhance the text markup language. CSS2 has the following most important features:

- Aural Style Sheets: New style properties for defining the aural style sheet for documents.
- **Paging:** Definition of how pages need to be displayed or printed. This made cropping, registering marks and other layout features possible.
- **Media types:** A different style rule for different types of media was introduced in CSS2.
- **International accessibility features:** More list styles were available for international documents. This included bidirectional text support as well as language sensitive quotation marks.
- **Font:** More fonts were defined and available for use.
- **Positioning:** CSS2 introduced the relative, absolute positioning and the placement determination within a document. This really helped the continuous media.
- **Cursors:** CSS2 defined the manner in which the cursor would respond to various actions.

### 3.9.2. Overview and Features of CSS3

**CSS3** is the latest development of the Cascading Style Sheets language and aspire at extending CSS2.1. It brings a lot of new features, like rounded corners, shadows, gradients, transitions or animations, in addition to new layouts like multi-columns, flexible box or grid layouts.

#### 3.9.2.1. CSS3 Features

In CSS3, there are many options, a few which we'll look at here. These comprise animation, gradients, media queries, shadows, transitions, the font-face rule that allows you to embed fonts on a web page, and more.

1. **Animations:** by this option you can add key frame animations to HTML content. With this approach you can move from one CSS style to another. There are two main components in a CSS animation: A style that describes and the key frames that control the animation. There are three reasons for using CSS animations instead of animations run by scripts. These are:
  1. These are easy to use and can be done without a knowledge of JavaScript.
  2. They run well under a moderate system load.



3. By letting the browser control the animation, the browser can optimize the performance.
2. **Gradients:** In CSS3, there's more support for gradients with the `<image>` tag. This property allows you to display smooth transitions between two or more colors using the browser instead of images. This reduces download times. The browser supports two types of gradients: Linear and Radial. The linear gradient is defined with the linear-gradient function, while radial is defined with the radial-gradient function.
3. **CSS3 media queries:** Mobile devices have dramatically changed the way that people function. All have different dimensions and web designers are faced with the challenge of making sure that their websites display properly on each device. Media queries are designed to handle this issue. Media queries build on the CSS2 property. They allow for multiple rule sets which are based on the capabilities of the browser. These include height and width controls, screen orientation and resolution. This gives you greater control over how your content is displayed and allows the designer to optimize the layout and design for different devices.
4. **CSS3 shadows:** With this property you can add shadows to HTML content. The box-shadow property allows you to add shadows to both the inner and outer box elements and the text-shadow property gives you control of text shadows.
5. **CSS3 transitions:** With CSS3 transitions you can animate CSS properties HTML elements. Some applications of this are: animated rollovers, fades, changing color from white to black, and more. Most importantly, JavaScript isn't necessary. One caveat is that while it's easier to use CSS3 transitions, you don't have as much control as you would with CSS3 animations.
6. **CSS3 transforms:** With CSS3 transforms, you can use rotations, scales and skews to DOM elements. CSS3 transforms can be used with CSS3 transitions, allowing you to create time controlled animations. Two main properties control CSS3 transforms. These are: transform and transform-origin. Transform covers the changes applied to the element, which happen one after the other. Transform-origin covers the origin position, usually the top left corner of the element, though this can be moved.



## CHAPTER 4

# BASIC OF JAVASCRIPT

### CONTENTS

4.1. Clients-Side Javascript .....	108
4.2. Data Types and Variables in Javascript.....	109
4.3. Functions in Javascript .....	112
4.4. Conditions Statement in Javascript .....	116
4.5. Javascript Loops .....	122
4.6. Javascript Pop Up Boxes .....	127
4.7. Advance Javascript: Javascript and Objects.....	132
4.8. DOM.....	138
4.9. Javascript form Validation.....	139
4.10. DHTML .....	141

JavaScript is a dynamic web programming language for the webpages. It is usually used as an element of web pages, whose implementations permit in the client-side script to interact with the user and make dynamic web pages. JavaScript is an interpreted programming language with an object-oriented facility. JavaScript was first recognized as LiveScript, but Netscape changed its name to JavaScript, because of the excitement being generated by Java. JavaScript prepared its first appearance in Netscape 2.0 in 1995 with the name LiveScript. The general-purpose foundation of the language has been embedded in Netscape, internet explorer, and other web browsers.

## 4.1. CLIENTS-SIDE JAVASCRIPT

Client-side JavaScript is the most familiar appearance of the web language. The script should be integrated or referenced by an HTML manuscript, and the code to be interpreted by the browser. It means a web page should not be a static HTML, but it can include programs or script code that interact with the user, manage the browser, and dynamically create HTML file content. The client-side JavaScript method provides many advantages over traditional CGI server-side scripts program. The JavaScript code is executed after submits the form by the user, and only if all the form entries are valid, they would be submitted to the Web Server. JavaScript can be used to catch user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

### 4.2.1. Advantages of JavaScript

The qualities of using JavaScript are:

- **Less server interaction:** JavaScript can validate user input before sending the page off to the server. These save server traffic, which means fewer loads on your server.
- **Instant feedback to the visitors:** they don't wait for a page reload to observe if they have forgotten to enter something.
- **Better interactivity:** you can create interfaces that respond when the user float over them with a mouse or activates them via the keyboard.
- **Richer interfaces:** you can use JavaScript to include such items as drag-and-drop mechanism and sliders to give a Rich Interface to your website visitors.

### 4.2.2. Limitations of JavaScript

We cannot delight JavaScript as a full-fledged programming language. It is short of the following important features:

- Client-side JavaScript does not permit the reading or writing of files. This has been set aside for security basis.
- JavaScript cannot be used for networking applications because there is no such type support existing.
- JavaScript doesn't have any multi-threading or multiprocessor facility.

### 4.2.3. Syntax of JavaScript

JavaScript can be executed with JavaScript statements that are placed within the `<script>`, `</script>` HTML tags in a web page. You can put the `<script>` tags, containing your JavaScript, anywhere within your web page, but it is generally suggested that you should keep it within the `<head>` tags.

Syntax:

```
<script.>
```

JavaScript code

```
</script>
```

The script tags take two important features:

- **Language** – this feature specifies what scripting language you are using. Although latest versions of HTML (and XHTML) have phased out the use of this quality.
- **Type** – this feature is recommended to indicate the scripting language in use and its value should be set to “text/JavaScript.”

## 4.2. DATA TYPES AND VARIABLES IN JAVASCRIPT

1. **JavaScript data types:** One of the most essential characteristics of a programming language is set of data types for it supports. These are the type of values that can be represented and operated in a programming language. JavaScript permits you to work with three primitive data types:
  - **Numbers**, e.g., 124, 121.50 etc.
  - **Strings** of text, e.g., “This is a text string,” etc.

- **Boolean**, e.g., true or false.

JavaScript also classify two minor data types, **null** and **undefined**, each of which defines only a single value. These primitive data types, supports a composite data type known as **object**.

2. **JavaScript variables:** similar to other programming languages, JavaScript has variables. Variables can be thought of as named containers; you can place data into these containers and then refer to the data simply by naming the container. Variables are declared with as follow the **var** keyword.

```
<script type = "text/javascript">
<!--
var cost;
var name;
//-->
</script>
```

With the same **var** keyword, can also declare multiple variables as follows:

```
<script type = "text/javascript">
<!--
var cost, name;
//-->
</script>
```

Store a value in a variable is called variable initialization. You can perform variable initialization at the time of variable creation or at a later point in time when you need that variable. For example, you may create variable named cost and assign the value 2000.50 to it later.

```
<script type = "text/javascript">
<!--
var name = "Aman";
var cost;
money = 2000.50;
//-->
</script>
```

3. **JavaScript variable scope:** The scope of a variable is the area of your program in which it is defined. JavaScript variables have only two scopes.
  - i. **Global variables:** A global variable has global scope which means it can be defined anywhere in your JavaScript code.
  - ii. **Local variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.
4. **JavaScript variable names:** as your variables name in JavaScript, keep the following rules in your mind:
  - The JavaScript reserved keywords should not use as a variable name. Such as, **break** or **Boolean** are not valid variable names.
  - The JavaScript variable names should not start with a numeral (0–9). They should begin with a letter or an underscore character. Such as, **123test** is an invalid variable name but **\_123test** is a valid one.
  - The JavaScript variable names are case-sensitive. Such as, **Name** and **name** are two different variables.
5. JavaScript Reserved Words:

A list of the reserved keywords in JavaScript is given; they cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

**Table 4.1.** A list of the reserved keywords in JavaScript

abstract	else	Instance of	switch
Boolean	Enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	type of
continue	function	protected	var
debugger	go to	public	void

default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

### 4.3. FUNCTIONS IN JAVASCRIPT

A function is a collection of reusable program or code which can be called anywhere in your program. These eliminate the need of writing the same code again and again. Functions allow a programmer to divide a large program into a number of small and convenient functions.

**1. Function definition:** The function can be define in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

**Syntax:**

```
<script type = "text/javascript">
<!--
function functionname(parameter-list) {
statements
}
//-->
</script>
```

**Example:** It defines a function called sayHello that takes no parameters:

```
<script type = "text/javascript">
<!--
function sayHello() {
alert("Hello there");
}
//-->
</script>
```

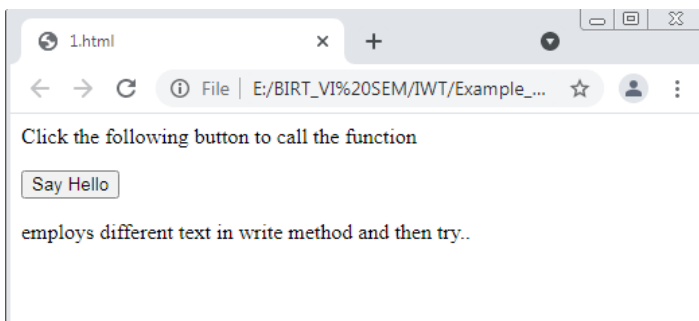
**2. Calling a Function:** A function invoke in the script; you only write the name of that function.



**Example: Call a Function 4.1**

```
<html>
<head>
<script type = "text/javascript">
function sayHello() {
document. write ("Hello there!");
}
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type = "button" onclick = "sayHello()" value = "Say Hello">
</form>
<p>employs different text in write method and then try.</p>
</body>
</html>
```

Output:



**Figure 4.1.** *Output of call a function.*

- Parameters in Function:** Previously, we have seen an example of functions without parameters. But there is a capability to pass different parameters at the same time as calling a function. These passed parameters can be captured within the function and any manipulation can be done over those parameters. A function can

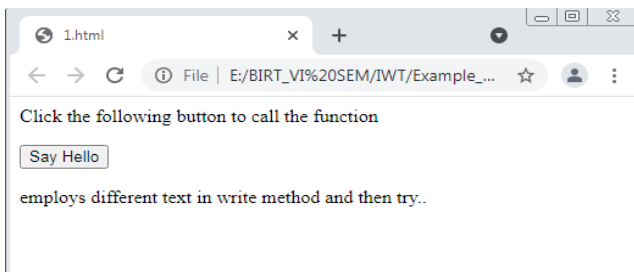
be taking multiple parameters separated by comma. We have modified previous define sayHello function, at this time it takes two parameters.

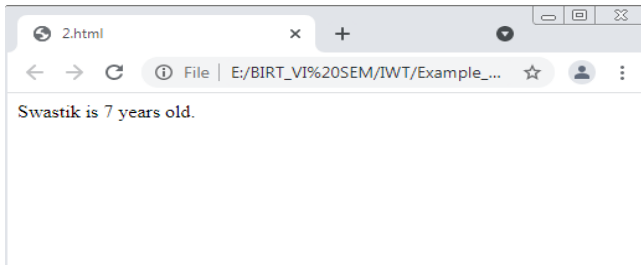
#### Example: Parameters Function 4.2

```
<html>
<head>
<script type = "text/javascript">
function sayHello(name, age) {
document. write (name + " is " + age + " years old.");
}
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type = "button" onclick = "sayHello('Swastik,' 7)" value = "Say
Hello">
</form>
<p>employs different parameters inside the function and then try.</p>
</body>
</html>
```

Output:

After clicked say Hello button output displayed as figure 4.2.





**Figure 4.2.** *Output of parameters function.*

- 4. The return statement:** A JavaScript function is able to have an optional return statement. This is required when you want to return a value from a function. This statement should be use as the last statement in a function. In Example defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

#### **Example: return\_function 4.3**

```
<html>
<head>
<script type = "text/javascript">
function concatenate(first, last) {
var full;
full = first + last;
return full;
}
function secondFunction() {
var result;
result = concatenate('Swastik,' 'Aman');
document. write (result);
}
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
```

```
<input type = "button" onclick = "secondFunction()" value = "Call  
Function">
```

```
</form>
```

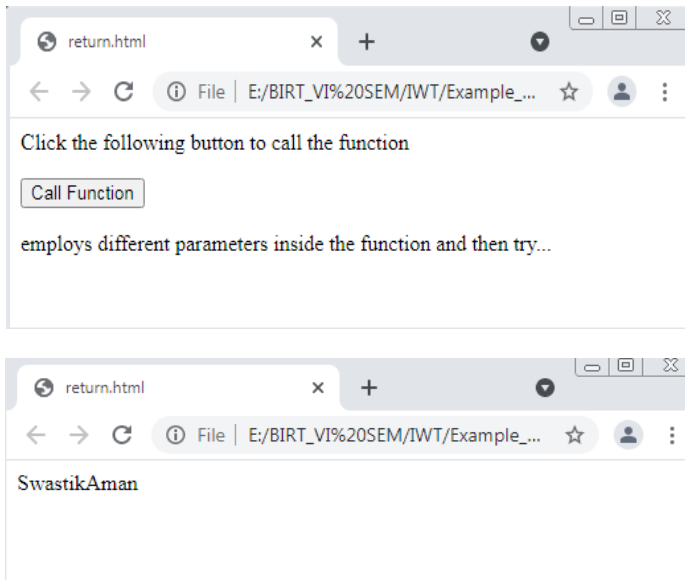
```
<p>employs different parameters inside the function and then try.</p>
```

```
</body>
```

```
</html>
```

Output:

After clicked say Hello button output displayed as Figure 4.3.



**Figure 4.3.** *Output of return function.*

## 4.4. CONDITIONS STATEMENT IN JAVASCRIPT

Conditional statements are used to perform special actions based on conditions. In JavaScript we have the following different conditional statements:

- **if** block of code to be executed, if a particular condition is true.
- **else** block of code to be executed, if the same condition is false.
- **else if** block of code specify a new condition to test, if the first condition is false.

- **switch** block of code specify many alternative blocks of code to be executed.
1. **If statement:** Block of **if** statement to specify a block of JavaScript code to be executed if a condition is true.

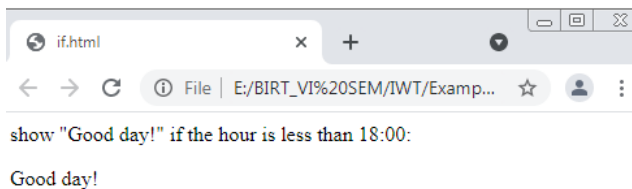
**Syntax:**

```
if (condition) {
// block of code executed if the condition is true
}
```

**Example: if\_statement 4.4**

```
<html>
<body>
<p>show "Good day!" if the hour is less than 18:00:</p>
<p id="demo">Good Evening! </p>
<script>
if (new Date().getHours() < 18) {
document.getElementById("demo").innerHTML = "Good day!";
}
</script>
</body>
</html>
```

Output:

**Figure 4.4.** *Output of if statement.*

2. **Else statement:** Else statement to specify a block of code to be executed if the condition is false.

Syntax:

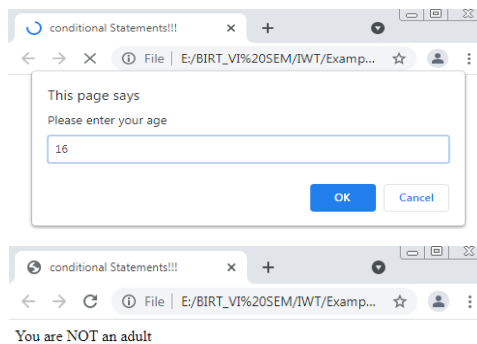
```
if (condition) {  
  // block of code to be executed if the condition is true  
} else {  
  // block of code to be executed if the condition is false  
}
```

**Example: ifelse\_4.5**

```
<html>  
<head>  
<title>conditional Statements!!!</title>  
<script type="text/javascript">  
var age = prompt("Please enter your age");  
if(age>=18)  
document.write("You are an adult <br />");  
else  
document.write("You are NOT an adult <br />");  
</script></head>  
<body></body>  
</html>
```

Output:

After clicked ok button output displayed as Figure 4.5



**Figure 4.5.** Output of if\_else statement.

**3. The else if statement:** else if statement to specify a new condition if the first condition is false.

**Syntax:**

```

if (condition1) {
    // block of code to be executed if condition1 is true
} else if (condition2) {
    // block of code to be executed if the condition1 is false and condition2
    is true
} else {
    // block of code to be executed if the condition1 is false and condition2
    is false
}

```

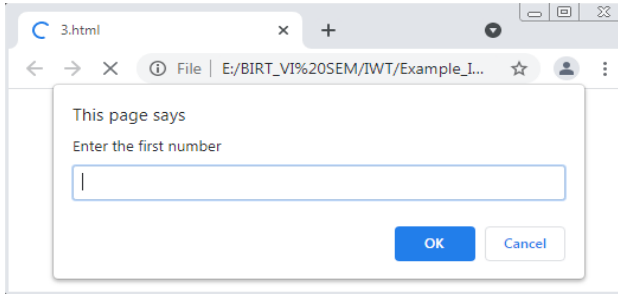
**Example: elseif\_4.6**

```

<html>
<head>
<script type="text/javascript">
var one = prompt("Enter the first number");
var two = prompt("Enter the second number");
one = parseInt(one);
two = parseInt(two);
if (one == two)
document.write(one + " is equal to" + two + ".");
else if (one<two)
document.write(one + " is less than" + two + ".");
else
document.write(one + " is greater than" + two + ".");
</script>
</head>
<body>
</body>
</html>

```

Output:



**Figure 4.6.** *Output of elseif statement.*

**4. JavaScript switch statement:** The switch statement is used to specify many alternative blocks of code to be executed different actions based on different conditions.

**Syntax:**

```
switch(expression) {  
  case x:  
    // code block  
    break;  
  case y:  
    // code block  
    break;  
  default:  
    // code block  
}
```

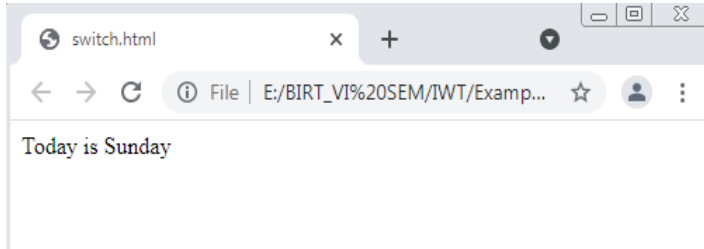
- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.
- If there is no match, the default code block is executed.
- The default keyword specifies the code to run if there is no case match
- When JavaScript reaches a break keyword, it breaks out of the switch block.



**Example: switch\_4.7**

```
<html>
<body>
<p id="demo"></p>
<script>
var day;
switch (new Date(). getDay()) {
case 0:
day = "Sunday";
break;
case 1:
day = "Monday";
break;
case 2:
day = "Tuesday";
break;
case 3:
day = "Wednesday";
break;
case 4:
day = "Thursday";
break;
case 5:
day = "Friday";
break;
case 6:
day = "Saturday"; }
document.getElementById("demo").innerHTML = "Today is" + day;
</script>
</body>
</html>
```

Output:



**Figure 4.7.** *Output of switch statement.*

## 4.5. JAVASCRIPT LOOPS

A loop is a block of code can execute a number of times. If you want to run the same code frequently, each time with a different value. JavaScript supports different type of loops:

- for – loop through a block of code execute a number of times
- for/in – loops through the properties of an object
- for/of – loops through the values of an iterable object
- while – loops through a block of code while a specified condition is true
- do/while – also loops through a block of code while a specified condition is true

**1. For loop:** for loop has the following syntax:

for (statement 1; statement 2; statement 3)

```
{  
  // code block to be executed  
}
```

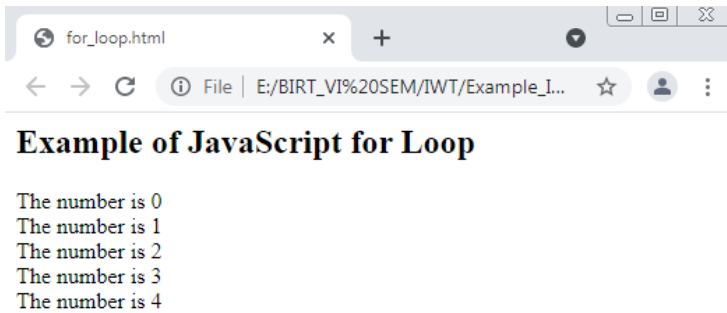
- Statement 1 is executed (one time) before the execution of the code block.
- Statement 2 defines the condition for executing the code block.
- Statement 3 is executed (every time) after the code block has been executed.

Example: for\_loop 4.8

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

```
<h2>Example of JavaScript for Loop</h2>
<p id="demo"></p>
<script>
var text = "";
var i;
for (i = 0; i < 5; i++) {
text += "The number is " + i + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

Output:



**Figure 4.8.** *Output of for loop.*

- 2. The For/In Loop:** The JavaScript for/in statement loops executed from beginning to end the properties of an object.

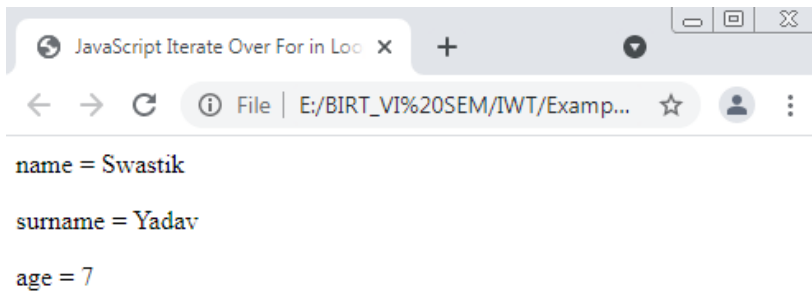
Syntax:

```
for (variablename in object) {
statement or block to execute
}
```

Example: for-in 4.9

```
<html >
<head>
```

```
<title>JavaScript Iterate Over For in Loop</title>
</head>
<body>
<script>
  var person = {"name": "Swastik," "surname": "Yadav," "age": "7"}; //
An object with properties
  for(var prop in person) {
    document. write("<p>" + prop + " = " + person[prop] + "</p>"); //
properties in the object
  }
</script>
</body>
</html>
```

**Output:**

**Figure 4.9.** *Output of for-in loop.*

- 3. For/Of Loop:** JavaScript for/of statement loops through the values of iterable objects. For/of loop over data structures that are iterable such as Arrays, Strings, Maps, NodeLists, and more.

**Syntax:**

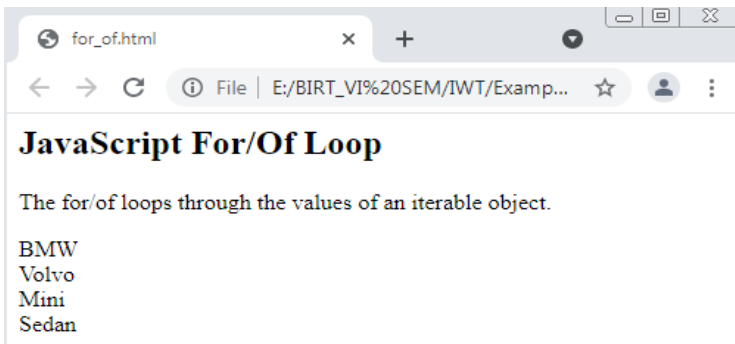
```
for (variable of iterable)
{
  // code block to be executed
}
```

**Example: for\_of 4.10**

```
<html>
```

```
<body>
<h2>JavaScript For/Of Loop</h2>
<p>The for/of loops through the values of an iterable object.</p>
<script>
var cars = ["BMW," "Volvo," "Mini," "Sedan"];
var x;
for (x of cars) {
document.write(x + "<br >");
}
</script>
</body>
</html>
```

Output:



**Figure 4.10.** *Output of for-of loop.*

4. **While loop:** loop through a block of code while a specified condition is true.

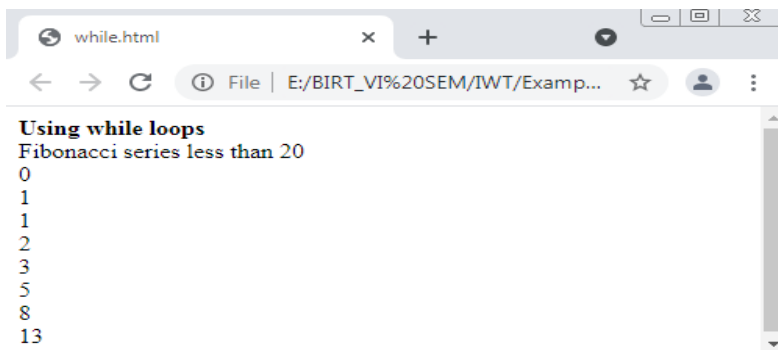
**Syntax:**

```
while (condition) {
// code block to be executed}
```

**Example: while 4.11**

```
<html>
<head><script type="text/javascript">
document.write("<b>Using while loops </b><br />");
```

```
var i = 0, j = 1, k;  
document.write("Fibonacci series less than 20<br />");  
while(i<20)  
{ document.write(i + "<br />");  
k = i+j;  
i = j;  
j = k;}  
</script></head>  
<body>  
</body>  
</html>
```

**Output:**

**Figure 4.11.** *Output of while loop.*

5. **Do/while loop:** The do/while loop is a variation of the while loop. Loop will execute the code block once, before checking if the condition is true, and then it will repeat the loop since the condition is true.

**Syntax:**

```
do {  
  // code block to be executed  
}
```

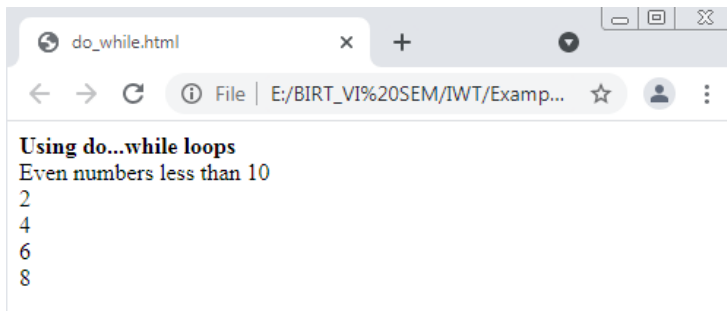
```
while (condition);
```

**Example: do-while 4.12**

```
html>
```

```
<head>
<script type="text/javascript">
document.write("<b>Using do.while loops </b><br />");
var i = 2;
document.write("Even numbers less than 10<br />");
do
{document.write(i + "<br />");
i = i + 2;
} while(i<10)
</script>
</head>
<body>
</body>
</html>
```

Output:



**Figure 4.12.** *Output of do-while loop.*

## 4.6. JAVASCRIPT POP UP BOXES

JavaScript has three types of popup boxes: Alert box, Confirm box, and Prompt box.

1. **Alert box:** An alert box is frequently used if you want to make sure information comes through to the user. While an alert box pops up, the user will have to click “OK” to proceed.

Syntax:

```
window.alert("sometext");
```

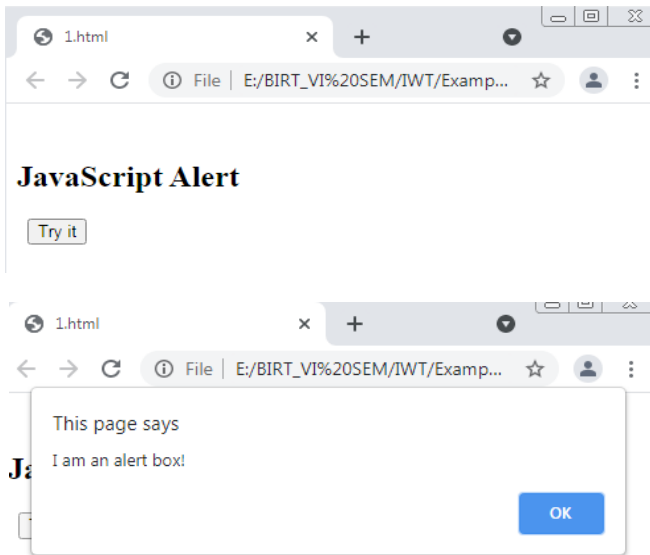
The `window.alert()` method can be written without the `window` prefix.

**Example: alert 4.13**

```
<html>
<body>
<h2>JavaScript Alert</h2>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction() {
  alert("I am an alert box!");
}
</script>
</body>
</html>
```

**Output:**

After clicked try it button output displayed as Figure 4.13



**Figure 4.13.** *Output of alert popup.*



2. **Confirm popup box:** A Confirm Popup box is frequently used if you want the user to verify. As a confirm box pops up, the user will have to click either “OK” or “Cancel” to proceed. If the user clicks “OK,” the box returns true. If the user clicks “Cancel,” the box returns false.

**Syntax:**

```
window.confirm(“sometext”);
```

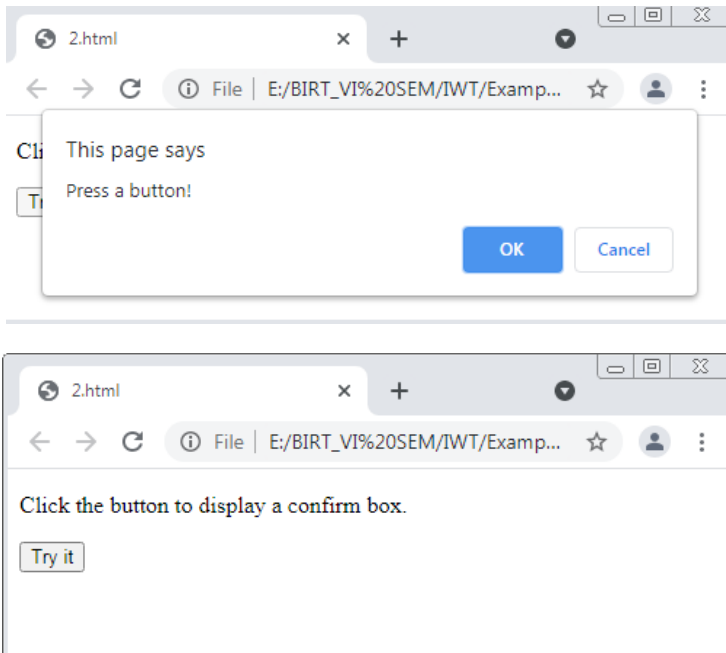
The window. confirm () method can be written without the window prefix.

**Example: confirmpop 4.14**

```
<html>
<body>
<h2>JavaScript Confirm Box</h2>
<button onclick=“myFunction()”>Try it</button>
<p id=“demo”></p>
<script>
function myFunction()
{
var txt;
if (confirm(“Press a button!"))
{
txt = “You pressed OK!”;
}
else {
txt = “You pressed Cancel!”;
}
document.getElementById(“demo”).innerHTML = txt;
}
</script>
</body>
</html>
```

**Output:**

After clicked Try it button output displayed as Figure 4.14



**Figure 4.14.** *Output of confirm popup.*

3. **Prompt popup box:** A prompt popup box is frequently used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either “OK” or “Cancel” to proceed after entering an input value. If the user clicks “OK” the box returns the input value. If the user clicks “Cancel” the box returns null.

Syntax:

```
window. prompt(“sometext,” “defaultText”);
```

The window. prompt() method can be written without the window prefix.

Example: prompt 4.15

```
<html>
```

```
<body>
```

```
<h2>JavaScript Prompt</h2>
```

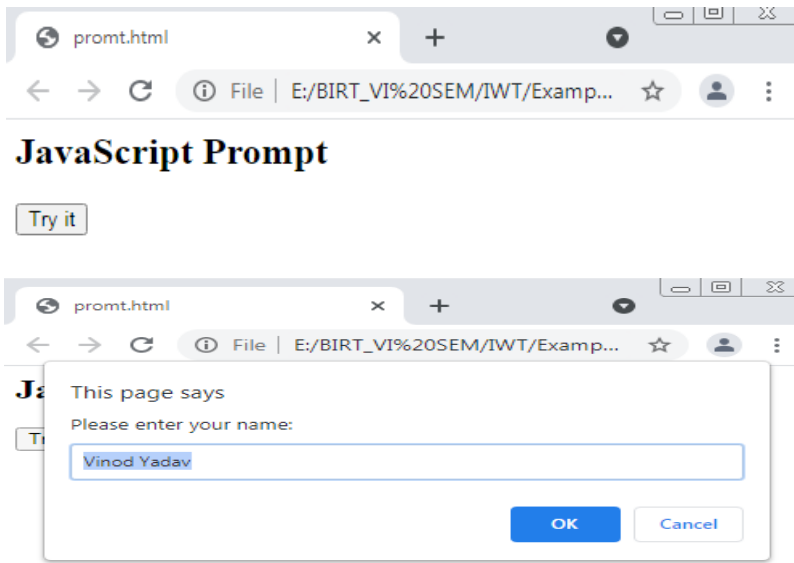
```
<button onclick=“myFunction()”>Try it</button>
```

```
<p id=“demo”></p>
```

```
<script>
function myFunction() {
var txt;
var person = prompt("Please enter your name:," "Vinod Yadav");
if (person == null || person == "") {
txt = "User canceled the prompt.";
} else {
txt = "Hello " + person + "! How are you today?";
}
document.getElementById("demo").innerHTML = txt;
}
</script>
</body>
</html>
```

**Output:**

After clicked try it button output displayed as Figure 4.15.



**Figure 4.15.** *Output of prompt popup.*

## 4.7. ADVANCE JAVASCRIPT: JAVASCRIPT AND OBJECTS

Objects, in JavaScript, are its most essential data-type and form the building blocks for current JavaScript. These objects are quite different from JavaScript's primitive data-types (Number, String, Boolean, null and symbol) in the sense that while these primitive data-types all store a single value each (depending on their types).

- Objects are more complex and each object may include any combination of these primitive data-types as well as reference data-types.
- An object is a reference data type. Variables that are assigned a reference value are given a reference or a pointer to that value. That reference or pointer points to the location in memory where the object is stored. The variables don't actually store the value.

An object can be created with figure brackets {...} with an optional list of properties. A property is a "key: value" pair, where a key is a string (also called a "property name"), and value can be anything. For Example:

```
let school = {  
  name: "Vivekananda High School,"  
  location: "Bhopal,"  
  established: "1981"  
}
```

In the above example "name," "location," "established" are all "keys" and "Vivekananda High School," "Bhopal" and 1981 are values of these keys in that order. All of these keys are referred to as properties of the object. An object in JavaScript may also have a function as a member, in which case it will be known as a method of that object.

1. **Object properties:** JavaScript Object properties can be any of the three primitive data types, or any of the abstract data types. Object properties are generally variables that are used internally in the object's methods, but can also be globally visible variables that are used all over the page.

Syntax for adding a property to an object:

```
objectName.objectProperty = propertyValue;
```

**For example:** code gets the document title using the “**title**” property of the **document** object.

```
var str = document. title;
```

2. **Object methods:** the functions that allow the object do something or allow something be done to it. There is a small difference between a function and a method, at a function is a standalone unit of statements and a method is attached to an object and can be referenced by **this** keyword.

**For example:** it show how to use the **write ()** method of document object to write any content on the document.

```
document. write(“This is test”);
```

3. **User-defined objects:** every user-defined objects and built-in objects are descendants of an object called Object.
4. **The new operator:** new operator is used to form an instance of an object. To create an object, the new operator is followed by the constructor method. The constructor methods are Object (), Array (), and Date (). These constructors are built-in JavaScript functions.

### Example:

```
var employee =newObject();
```

```
var books =newArray(“C++”,“Perl”,“Java”);
```

```
var day =newDate(“August 15, 1947”);
```

5. **The object () constructor:** A constructor is a function that uses to create and initializes an object. JavaScript provides a particular constructor function called Object () to build the object. The return value of the Object () constructor is assigned to a variable. The variable contains a reference to the new object.

Example: object\_constructor 4.16

```
<html>
```

```
<head>
```

```
<title>User-defined objects</title>
```

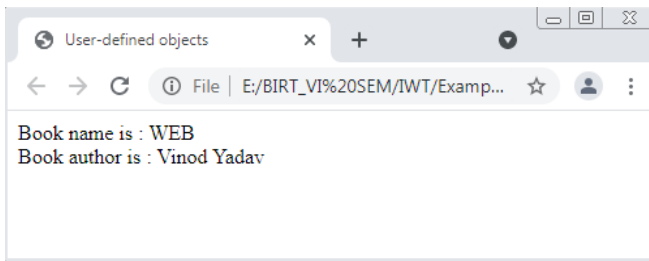
```
<script type = “text/javascript”>
```

```
var book = new Object(); // Create an object
```

```
book.subject = “WEB”; // Assign properties to the object
```

```
book.author = “Vinod Yadav”;
```

```
</script>
</head>
<body>
<script type = "text/javascript">
document.write("Book name is: " + book.subject + "<br>");
document.write("Book author is: " + book.author + "<br>");
</script>
</body>
</html>
```

**Output:**

**Figure 4.16.** *Output of object.*

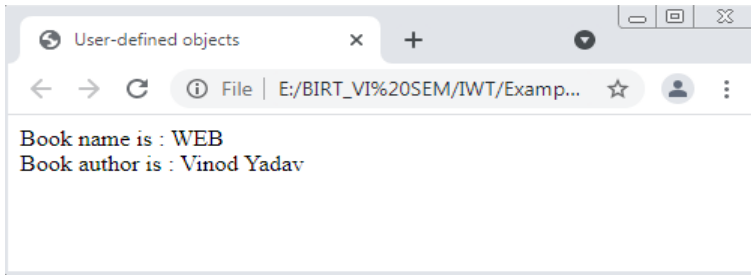
**Example: this 4.17**

```
<html>
<head>
<title>User-defined objects</title>
<script type = "text/javascript">
function book(title, author) {
this.title = title;
this.author = author;
}
</script>
</head>
<body>
<script type = "text/javascript">
var myBook = new book("WEB," "Vinod Yadav");
document.write("Book title is: " + myBook.title + "<br>");
document.write("Book author is: " + myBook.author + "<br>");
```

```
</script>
```

```
</body>
```

```
</html>
```

**Output:**

**Figure 4.17.** *Output of object.*

- 6. Defining methods for an object:** The earlier examples show how the constructor creates object and assigns properties. But we need to whole definition of an object by assigning methods to it.

**Example: methods 4.18**

```
<html>
```

```
<head>
```

```
<title>Methods for an objects</title>
```

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

```
function addPrice(amount){
```

```
  this. price = amount;
```

```
}
```

```
function book(title, author)
```

```
{
```

```
  this.title = title;
```

```
  this.author = author;
```

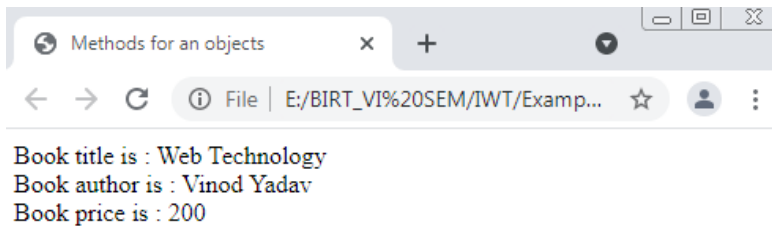
```
  this.addPrice = addPrice; // Assign that method as property.
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
<script type="text/javascript">
var myBook =new book("Web Technology," "Vinod Yadav");
myBook.addPrice(200);
document.write("Book title is: "+ myBook.title + "<br>");
document.write("Book author is: "+ myBook.author + "<br>");
document.write("Book price is: "+ myBook.price + "<br>");
</script>
</body>
</html>
```

**Output:**

**Figure 4.18.** *Output of methods for object.*

## 7. Build-in JavaScript Object Methods

A variety of methods of Object are as follows:



Table 4.2. A variety of methods of object

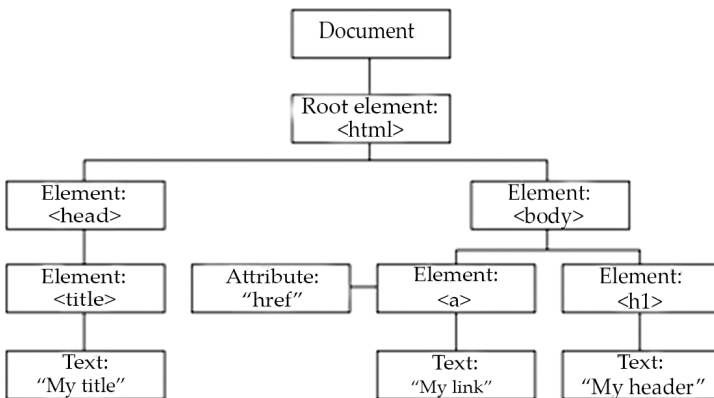
S. No	Methods	Description
1	Object.assign()	This method is used to copy enumerable and own properties from a source object to a target object
2	Object.create()	This method is used to create a new object with the specified prototype object and properties.
3	Object.defineProperty()	This method is used to describe some behavioral attributes of the property.
4	Object.defineProperties()	This method is used to create or configure multiple object properties.
5	Object.entries()	This method returns an array with arrays of the key, value pairs.
6	Object.freeze()	This method prevents existing properties from being removed.
7	Object.getOwnPropertyDescriptor()	This method returns a property descriptor for the specified property of the specified object.
8	Object.getOwnPropertyDescriptors()	This method returns all own property descriptors of a given object.
9	Object.getOwnPropertyNames()	This method returns an array of all properties (enumerable or not) found.
10	Object.getOwnPropertySymbols()	This method returns an array of all own symbol key properties.
11	Object.getPrototypeOf()	This method returns the prototype of the specified object.
12	Object.is()	This method determines whether two values are the same value.
13	Object.isExtensible()	This method determines if an object is extensible
14	Object.isFrozen()	This method determines if an object was frozen.
15	Object.isSealed()	This method determines if an object is sealed.

16	Object.keys()	This method returns an array of a given object's own property names.
17	Object.preventExtensions()	This method is used to prevent any extensions of an object.
18	Object.seal()	This method prevents new properties from being added and marks all existing properties as non-configurable.
19	Object.setPrototypeOf()	This method sets the prototype of a specified object to another object.
20	Object.values()	This method returns an array of values.

## 4.8. DOM

The document object model (DOM) is an application programming interface (API) for HTML and XML documents. The DOM is a world wide web consortium (W3C) standard. The DOM defines a standard for accessing documents:

“The W3C DOM is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.” The **HTML DOM** model is constructed as a tree of **Objects**:



**Figure 4.19.** *HTML DOM.*

1. **HTML DOM:** The HTML DOM is a standard object model and programming interface for HTML. It defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements

The Objects are organized in a hierarchical structure applies to the organization of objects in a Web document.

- **Window object:** Top of the hierarchy. It is the outmost element of the object hierarchy.
- **Document object:** Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.
- **Form object:** everything enclosed in the <form>.</form> tags sets the form object.
- **Form control elements:** the form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

## 4.9. JAVASCRIPT FORM VALIDATION

It is validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user. JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation. With JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

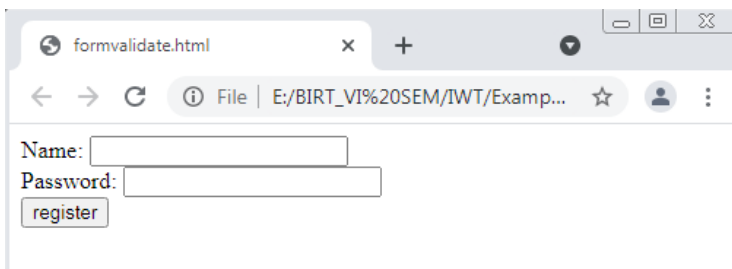
### 4.9.1. JavaScript Form Validation Example

In this example, we are going to validate the name and password. The name can't be empty and password can't be less than 6 characters long. Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

**Example: form validate 4.19**

```
<html>
<body>
<script>
```

```
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;
if (name==null || name==""){
alert("Name can't be blank");
return false;
}else if(password.length<6){
alert("Password must be at least 6 characters long.");
return false;
}
}
</script>
<body>
<form name="myform" method="post" action=" valid.jsp"
onsubmit="return validateform()">
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
</body>
</html>
```

**Output:**

**Figure 4.20.** *Output form validate.*

## 4.10. DHTML

**DHTML** stands for Dynamic Hypertext Markup language, Dynamic HTML (DHTML) is not a markup or programming language but it is a term that combines the features of various web development technologies for creating the web pages dynamic and interactive.

Dynamic HTML: **DHTML consists of the following four components or languages:**

- HTML
- CSS
- JavaScript
- DOM.

HTML is a client-side markup language, which is a core component of the DHTML. It defines the structure of a web page with various defined basic elements or tags. CSS stands for Cascading Style Sheet, which allows the web users or developers for controlling the style and layout of the HTML elements on the web pages. JavaScript is a scripting language which is done on a client-side. The various browser supports JavaScript technology. DHTML uses the JavaScript technology for accessing, controlling, and manipulating the HTML elements. The statements in JavaScript are the commands which tell the browser for performing an action. DOM is the DOM. It is a w3c standard, which is a standard interface of programming for HTML. It is mainly used for defining the objects and properties of all elements in HTML.

Following are the uses of Dynamic HTML (DHTML):

- It is used for designing the animated and interactive web pages that are developed in real-time.
- DHTML helps users by animating the text and images in their documents.
- It allows the authors for adding the effects on their pages.
- It also allows the page authors for including the drop-down menus or rollover buttons.
- This term is also used to create various browser-based action games.
- It is also used to add the ticker on various websites, which needs to refresh their content automatically.

### 4.10.1. Features of DHTML

Following are the various characteristics or features of DHTML:

- Its simplest and main feature is that we can create the web page dynamically.
- **Dynamic Style** is a feature, which allows the users to alter the font, size, color, and content of a web page.
- It provides the facility for using the events, methods, and properties. And, also provides the feature of code reusability.
- It also provides the feature in browsers for data binding.
- Using DHTML, users can easily create dynamic fonts for their web sites or web pages.
- With the help of DHTML, users can easily change the tags and their properties.
- The web page functionality is enhanced because the DHTML uses low-bandwidth effect.

**Table 4.3.** Difference between HTML and DHTML

HTML (Hypertext Markup language)	DHTML (Dynamic Hypertext Markup language)
1. HTML is simply a markup language.	1. DHTML is not a language, but it is a set of technologies of web development.
2. It is used for developing and creating web pages.	2. It is used for creating and designing the animated and interactive web sites or pages.
3. This markup language creates static web pages.	3. This concept creates dynamic web pages.
4. It does not contain any server-side scripting code.	4. It may contain the code of server-side scripting.
5. The files of HTML are stored with the .html or .htm extension in a system.	5. The files of DHTML are stored with the DHTM extension in a system.
6. A simple page which is created by a user without using the scripts or styles called as an HTML page.	6. A page which is created by a user using the HTML, CSS, DOM, and JavaScript technologies called a DHTML page.

7. This markup language does not need database connectivity.	7. This concept needs database connectivity because it interacts with users.
--	--

#### 4.10.2. DHTML JavaScript

JavaScript can be included in HTML pages, which creates the content of the page as dynamic. We can easily type the JavaScript code within the `<head>` or `<body>` tag of a HTML page. If we want to add the external source file of JavaScript, we can easily add using the `<src>` attribute.

Following are the various examples, which describe how to use the JavaScript technology with the DHTML:

##### Document.write() Method

The `document.write()` method of JavaScript, writes the output to a web page. This example simply uses the **`document.write()`** method of JavaScript in the DHTML. In this example, we type the JavaScript code in the **`<body>`** tag.

Example: Dhtml 4.20

```
<HTML>
```

```
<head>
```

```
<title>
```

```
Method of a JavaScript
```

```
</title>
```

```
</head>
```

```
<body>
```

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

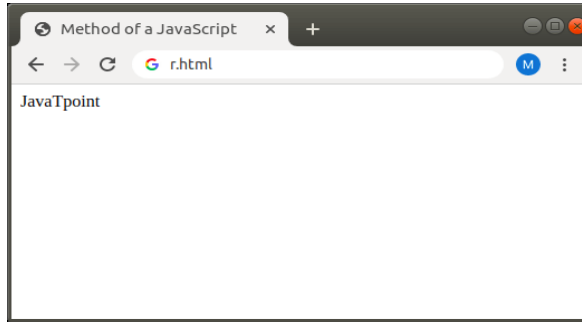
```
document.write("JavaTpoint");
```

```
</script>
```

```
</body>
```

```
</html>
```

Output:



**Figure 4.21.** *Output example 4.20.*

### 4.10.3. JavaScript and HTML Event

A JavaScript code can also be executed when some event occurs. Suppose, a user clicks an HTML element on a webpage, and after clicking, the JavaScript function associated with that HTML element is automatically invoked. And, then the statements in the function are performed. This example shows the current date and time with the JavaScript and HTML event. In this example, we type the JavaScript code in the <head> tag.

#### **Example: htmlevent 4.21**

```
<html>
<head>
<title>DHTML with JavaScript </title>
<script type="text/javascript">
function dateandtime()
{
alert(Date());
}
</script>
</head>
<body bgcolor="orange">
<font size="4" color="blue">
<center><p>
```

```
Click here # <a href="#" onClick="dateandtime();"> Date and Time </
```

a>



# to check the today's date and time.

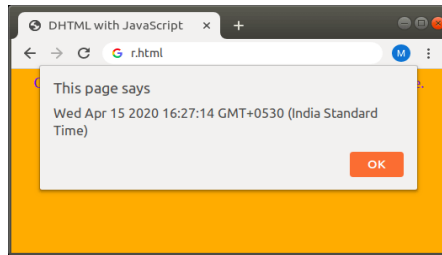
</p></center>

</font>

</body>

</html>

Output:



**Figure 4.22.** *Output example 4.21.*

JavaScript and HTML DOM example checks the Grade of a student according to the percentage criteria with the JavaScript and HTML DOM. In this example, we type the code of a JavaScript in the <body> tag.

### **Example: JavaScript and HTML DOM 4.22**

<html>

<head>

<title> Check Student Grade

</title>

</head>

<body>

<p>Enter the percentage of a Student:</p>

<input type="text" id="percentage">

<button type="button" onclick="checkGrade()">

Find Grade

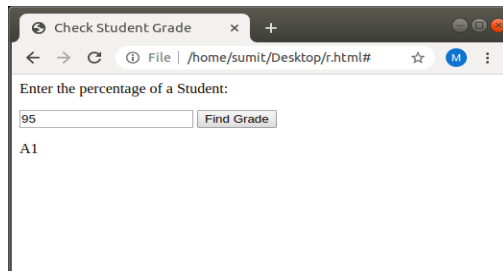
</button>

<p id="demo"></p>

<script type="text/javascript">

```
function checkGrade() {  
  var x,p, text;  
  p = document.getElementById("percentage").value;  
  x=parseInt(p);  
  if (x>90 && x <= 100) {  
    document.getElementById("demo").innerHTML = "A1";  
  } else if (x>80 && x <= 90) {  
    document.getElementById("demo").innerHTML = "A2";  
  } else if (x>70 && x <= 80) {  
    document.getElementById("demo").innerHTML = "A3";  
  }  
}  
  
</script>  
</body>  
</html>
```

Output:



**Figure 4.23.** *Output JavaScript and HTML DOM 4.22.*

### ***Advantages of DHTML***

Following are the various benefits or the advantages of DHTML:

- Those web sites and web pages which are created using this concept are fast.
- There is no plug-in required for creating the web page dynamically.

- Due to the low-bandwidth effect by the DHTML, the web page functionality is enhanced.
- This concept provides advanced functionalities than the static HTML.
- It is highly flexible, and the user can make changes easily in the web pages.

### ***Disadvantages of DHTML***

Following are the various disadvantages or limitations of DHTML:

- The scripts of DHTML do not run properly in various web browsers. Or in simple words, we can say that various web browsers do not support the DHTML. It is only supported by the latest browsers.
- The coding of those websites that are created using DHTML is long and complex.
- For understanding the DHTML, users must know about HTML, CSS, and JavaScript. If any user does not know these languages, then it is a time-consuming and long process in itself.



## **CHAPTER 5**

# **INTRODUCTION TO XML**

## **CONTENTS**

5.1. Uses of XML and Syntax of XML .....	150
5.2. XML DTD .....	154
5.3. XML Schema .....	158
5.4. Transforming XML Using XSL and XSLT.....	159

The extensible markup language referred to as XML. It is a text-based markup language that belongs from standard generalized markup language (SGML). XML tags recognize the data, which are used to store and organize the data. XML identify how to display it similar to HTML tags, which are used to display the data. XML is not going to change HTML in the near future, but it introduces new potential by adopting many successful features of HTML. There are three main characteristics of XML that make it useful in a variety of systems and solutions:

- **XML is extensible** – XML permits you to create your own or personal self-descriptive tags, that suits your particular application.
- **XML carries the data does not present it** – XML permits you to store the data irrespective of how it will be presented.
- **XML is a public standard** – XML was developed by an organization called the World Wide Web Consortium (W3C) and is presented as an open standard.

## 5.1. USES OF XML AND SYNTAX OF XML

Important XML usages are:

- XML can work behind the prospect to simplify the creation of HTML documents for large web sites.
- XML can be used to exchange the data or information between organization and system.
- XML can be used for offloading and reloading of databases.
- XML can be used to store and organize the data, which can customize your data handling needs.
- XML can be merged with style sheets to create almost any preferred output.

Example of XML: info.xml

```
<?xml version = "1.0"?>
<contact-info>
<name>Tanmay Patil</name>
<company>TutorialsPoint</company>
<phone>(011) 123-4567</phone>
</contact-info>
```

Output:

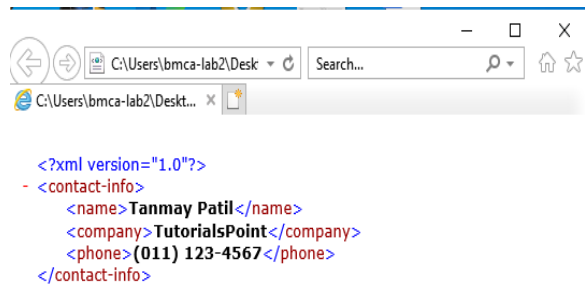


Figure 5.1. Output of info.xml.

5.1.1. Differences between XML and HTML

There are several differences between HTML and XML. Some important differences are given below:

Table 5.1. Differences between HTML and XML.

HTML	XML
HTML stands for <b>Hyper Text Markup Language</b> .	XML stands for <b>extensible Markup Language</b> .
HTML is static.	XML is dynamic.
HTML is a markup language.	XML provides framework to define markup languages.
HTML can ignore small errors.	XML does not allow errors.
HTML is not Case sensitive.	XML is Case sensitive.
HTML tags are predefined tags.	XML tags are user defined tags.
There are limited numbers of tags in HTML.	XML tags are extensible.
HTML does not preserve white spaces.	White space can be preserved in XML.
HTML tags are used for displaying the data.	XML tags are used for describing the data not for displaying.
In HTML, closing tags are not necessary.	In XML, closing tags are necessary.

1. **XML declaration or syntax:** The XML document written as follows:

```
<?xml version = "1.0" encoding = "UTF-8"?>
```

Where version is the XML version and encoding denote the character encoding used in the document.

2. **Rules for XML declaration:** The XML declaration is case sensitive and have to begin with "<?xml>" where "xml" is written in lower-case.

- The XML declaration strictly wants be the first statement in the XML document.
- An HTTP protocol can override the value of encoding that you set in the XML declaration.

3. **Tags and elements:** An XML file is prepared by several XML-elements, also called XML-nodes or XML-tags. The names of XML-elements are enclosed in triangular brackets <> as shown below:

```
<element>
```

4. **Syntax rules for tags and Elements:** Element Syntax: every XML-element wants to be closed either with start or with end elements as shown below:

```
<element>.</element>
```

- i. **Nesting of elements:** an XML-element can hold multiple XML-elements as its children, but the children elements must not have common characteristics. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

Incorrect nested tags Example:

```
<?xml version = "1.0"?>
```

```
<contact-info>
```

```
<company>XML Tutorials Point
```

```
</contact-info>
```

```
</company>
```

The Following example shows correct nested tags:

```
<?xml version = "1.0"?>
```

```
<contact-info>
```

```
<company>XML Tutorials Point</company>
```



<contact-info>

- ii. **Root element:** an XML document can contain only one root element. For example, following is not a correct XML document, for the reason that both the x and y elements occur at the top level without a root element –

<x>.</x>

<y>.</y>

The correct formed of XML document –

<root>

<x>.</x>

<y>.</y>

</root>

- iii. **Case sensitivity:** XML-elements are case-sensitive. That means the name of the start and the end elements should be exactly in the same case.

For example, <contact-info> is different from <Contact-Info>

- 5. **XML attributes:** An attribute specifies a single property for the element, using a name/value pair. An XML-element can have one or more attributes. For example:

<a href = “http://www.XMLtutorialspoint.com/”>XML Tutorialspoint! </a>

Here href is the attribute name and http://www.XMLtutorialspoint.com/ is attribute value.

- 6. **XML References:** References generally permit to add or include additional text or markup in an XML document. References at all times start with the symbol “&” which is a reserved character and end with the symbol “;”. XML has two types of references:

- i. **Entity references:** An entity reference have a name between the start and the end delimiters. For example: &amp; where amp is name. The name refers to a predefined string of text or markup.
- ii. **Character references:** A contain references, as &#65;, contains a hash mark (“#”) followed by a number. The number all the time refers to the Unicode code of a character. Here, 65 refer to alphabet “A.”

7. **XML text:** The names of XML-attributes and XML-elements are case-sensitive, which means the name of start and end elements need to be written in the same case. To avoid character encoding problems, all XML files should be saved as Unicode UTF-8 or UTF-16 files. Whitespace characters like blanks, tabs and line-breaks between XML-elements and XML-attributes will be ignored. Some characters are reserved by the XML syntax itself, they cannot be used directly. To use them, with the help of some replacement-entities, which are listed below:

Not Allowed Character	Replacement Entity	Character Description
<	&lt;	less than
>	&gt;	greater than
&	&amp;	ampersand
'	&apos;	apostrophe
"	&quot;	quotation mark

## 5.2. XML DTD

The XML Document Type Declaration, referred to as DTD, it is a technique to describe XML language precisely. DTDs make sure or verify vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language. An XML DTD can be specified either inside the document or outside the document (it can be kept in a separate document and then linked separately).

Basic syntax of a DTD is as follows:

**<! DOCTYPE element DTD identifier**

**[**  
**declaration1**  
declaration2

**.**  
**]>**

In the above syntax,

- The **DTD** starts with **<! DOCTYPE** delimiter.
- **Elements** inform the parser to parse the document from the

specified root element.

- **DTD identifier** is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called **External Subset**.
- **The square brackets [ ]** enclose an optional list of entity declarations called Internal Subset.

### 5.2.1. Internal DTD

A DTD is referred to as an internal DTD if elements are declared inside the XML files. To refer it as internal DTD, standalone attribute in XML declaration have to be set yes. This means, the declaration works independent of an external source.

#### Syntax

```
<! DOCTYPE root-element [element-declarations]>
```

Here root-element is the name of root element and element-declarations is where you declare the elements.

Example of internal DTD:

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address [
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
]>
<address>
<name>Aman Patil</name>
<company>XML TutorialsPoint</company>
<phone>(0775) 223-4567</phone>
</address>
```

In The above code:

**Start Declaration** – Begin the XML declaration with the following statement.

<?xml version = “1.0” encoding = “UTF-8” standalone = “yes” ?>

**DTD** – after the XML header, the *document type declaration* follows, it referred to as the DOCTYPE –

<! DOCTYPE address [

The DOCTYPE declaration has an exclamation mark (!) at the start of the element name. The DOCTYPE informs the parser that a DTD is associated with this XML document.

**DTD Body** – The DOCTYPE declaration is followed by the DTD body, where you declare elements, attributes, entities, and notations.

<!ELEMENT address (name, company, phone)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT company (#PCDATA)>

<!ELEMENT phone\_no (#PCDATA)>

Some elements are declared here that make the vocabulary of the <name> document. <! ELEMENT name (#PCDATA)> defines the element *name* to be of type “#PCDATA,” Here #PCDATA means parse-able text data.

**End Declaration** – finally, the declaration section of the DTD is closed using a closing bracket and a closing angle bracket (|>).

- **Basic Rules**
- The document type declaration must appear at the start of the document (preceded only by the XML header) – it is not permitted anywhere else within the document.
- Similar to the DOCTYPE declaration, the element declarations must start with an exclamation mark.
- The Name in the document type declaration must match the element type of the root element.

### 5.2.2. External DTD

In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal. *dtd* file or a valid URL. To refer it as external DTD, *standalone* attribute in the XML declaration must be set as **no**. This means, declaration includes information from the external source.

## Syntax

```
<!DOCTYPE root-element SYSTEM "file-name">
```

where *file-name* is the file with *dtd* extension.

### Example of External DTD

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>
```

```
<!DOCTYPE address SYSTEM "address.dtd">
```

```
<address>
```

```
<name>Aman Patil</name>
```

```
<company>XML TutorialPoint</company>
```

```
<phone>(0775) 223-4567</phone>
```

```
</address>
```

The content of the DTD file **address.dtd** is as shown:

```
<!ELEMENT address (name, company, phone)>
```

```
<!ELEMENT name (#PCDATA)>
```

```
<!ELEMENT company (#PCDATA)>
```

```
<!ELEMENT phone_no (#PCDATA)>
```

You can refer to an external DTD by using either **system identifiers** or **public identifiers**.

**System Identifiers:** A system identifier enables you to specify the location of an external file containing DTD declarations.

**Syntax:** `<!DOCTYPE name SYSTEM "address.dtd" [.]>`

As you can see, it contains keyword **SYSTEM** and a URI reference pointing to the location of the document.

**Public Identifiers:** Public identifiers provide a mechanism to locate DTD resources and are written as follows:

```
<!DOCTYPE name PUBLIC "-//Beginning XML//DTD Address Example//EN">
```

As you can see, it begins with keyword **PUBLIC**, followed by a specialized identifier. Public identifiers can follow any format; however, a commonly used format is called **Formal Public Identifiers, or FPIs**.

## 5.3. XML SCHEMA

XML Schema is generally known as XML schema definition (XSD). It is used to express and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

**Example shows how to use schema:**

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
  <xs:element name = "contact">
    <xs:complexType>
      <xs:sequence>
        <xs:element name = "name" type = "xs:string" />
        <xs:element name = "company" type = "xs:string" />
        <xs:element name = "phone" type = "xs:int" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

### 5.3.1. Definition Types

**Simple Type:** Simple type element is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date.

For example: <xs:element name = "phone\_number" type = "xs:int" />

**Complex Type:** A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example:

```
<xs:element name = "Address">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "name" type = "xs:string" />
```

```
<xs:element name = "company" type = "xs:string" />
<xs:element name = "phone" type = "xs:int" />
</xs:sequence>
</xs:complexType>
</xs:element>
```

In the above example, *Address* element consists of child elements. This is a container for other **<xs:element>** definitions, that allows to build a simple hierarchy of elements in the XML document.

### 5.3.2. Global Types

With the global type, you can define a single type in your document, which can be used by all other references. For example:

```
<xs:element name = "AddressType">
<xs:complexType>
<xs:sequence>
<xs:element name = "name" type = "xs:string" />
<xs:element name = "company" type = "xs:string" />
</xs:sequence>
</xs:complexType>
</xs:element>
```

## 5.4. TRANSFORMING XML USING XSL AND XSLT

Extensible stylesheet language (XSL) is a style language for XML and XSLT stands for XSL Transformations.

- XSLT is used to transform XML document from one form to another form.
- XSLT uses Xpath to perform matching of nodes to perform these transformations.
- The result of applying XSLT to XML document could be an XML document, HTML, text or any another document from technology perspective.
- The XSL code is written within the XML document with the extension of (*.xsl*).

- In other words, an XSLT document is a different kind of XML document.
1. **XML namespace:** XML Namespaces are the unique names.
    - XML Namespace is a mechanism by which element or attribute is assigned to a group.
    - XML Namespace is used to avoid the name conflicts in the XML document.
    - XML Namespace is recommended by W3C.
  2. XML namespace declaration: It is declared using reserved attribute such as the attribute is *xmlns* or it can begin with *xmlns:*  
Syntax: `<element xmlns:name = "URL">`

Where:

- Namespace starts with the *xmlns*.
- The word name is the namespace prefix.
- The URL is the namespace identifier.

**Example:** xml document named Table.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="rule. css"?>
<tables>
<table>
<tr>
<td>Apple</td>
<td>Banana</td>
</tr>
</table>
<table>
<height>100</height>
<width>150</width>
</table>
</tables>
```

In the above code, there would be a name conflict, both of them contains the same *table* element but the contents of the table element is different. To handle this situation, the concept of XML Namespace is used.



**Example:** XML document to resolve name conflict:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="rule.css"?>
<tables>
<m:table xmlns:m="http://www.google.co.in">
<m:tr>
<m:td>Apple</m:td>
<m:td>Banana</m:td>
</m:tr>
</m:table>
<n:table xmlns:m="http://www.yahoo.co.in">
<n:height>100</n:height>
<n:width>150</n:width>
</n:table>
</tables>
```

- **Xpath:**
  - Xpath is an important component of XSLT standard.
  - Xpath is used to traverse the element and attributes of an XML document.
  - Xpath uses different types of expression to retrieve relevant information from the XML document.
  - Xpath contains a library of standard functions.
- **Templates:**
  - An XSL stylesheet contains one or more set of rules that are called templates.
  - A template contains rules that are applied when the specific element is matched.
  - An XSLT document has the following things:
    - The root element of the stylesheet.
    - A file of extension.xml.
    - The syntax of XSLT i.e what is allowed and what is not allowed.

- The standard namespace whose URL is *http://www.w3.org/1999/XSL/Transform*.

**Example:** XML file that contains the information about five students and displaying the XML file using XSLT.

**XML file:** Students.xml as:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Rule.xml" ?>
<student>
<s>
<name> Divyank Singh Sikarwar </name>
<branch> CSE</branch>
<age>18</age>
<city> Agra </city>
</s>
<s>
<name> Aniket Chauhan </name>
<branch> CSE</branch>
<age> 20</age>
<city> Shahjahanpur </city>
</s>
<s>
<name> Simran Agarwal</name>
<branch> CSE</branch>
<age> 23</age>
<city> Buland Shar</city>
</s>
<s>
<name> Abhay Chauhan</name>
<branch> CSE</branch>
<age> 17</age>
<city> Shahjahanpur</city>
</s>
```

```
<s>
<name> Himanshu Bhatia</name>
<branch> IT</branch>
<age> 25</age>
<city> Indore</city>
</s>
</student>
```

**XSLT Code:** Creating Rule.xsl as:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h1 align="center">Students' Basic Details</h1>
<table border="3" align="center">
<tr>
<th>Name</th>
<th>Branch</th>
<th>Age</th>
<th>City</th>
</tr>
<xsl:for-each select="student/s">
<tr>
<td><xsl:value-of select="name"/></td>
<td><xsl:value-of select="branch"/></td>
<td><xsl:value-of select="age"/></td>
<td><xsl:value-of select="city"/></td>
</tr>
</xsl:for-each>
</table>
```

```
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

**Output:**

## Students' Basic Details

Name	Branch	Age	City
Divyank Singh Sikarwar	CSE	18	Agra
Ankit Chauhan	CSE	20	Shahjahanpur
Simran Agarwal	CSE	23	buland shar
Abhay Chauhan	CSE	17	Shahjahanpur
Himanshu Bhatia	IT	25	Indore

**Figure 5.2.** *Example of XSLT*

## CHAPTER 6

# INTRODUCTION TO PHP

## CONTENTS

6.1. General Uses of PHP .....	166
6.2. How to Run PHP Script .....	166
6.3. Basic Syntax of PHP.....	168
6.4. Php Conditional or Decision Statements.....	170
6.5. Php Loops.....	172
6.6. Php and HTML, Arrays, and Functions.....	175
6.7. PHP Strings.....	180
6.8. Form Processing and Files.....	182
6.9. Cookies and Sessions.....	187
6.10. Basic Object Oriented Programing In PHP.....	194

Hypertext Preprocessor referred to as PHP, it is a widely used open-source general purpose scripting language and it can be embedded with HTML code. PHP files are saved with “.php” extension. PHP scripts can be write anyplace in the document within PHP tags along with normal HTML tag.

- PHP is a server side scripting language that is embedded in HTML. It is used to control dynamic content, databases, session tracking, even build entire e-commerce sites.
- PHP is incorporated with a number of accepted databases, including MySQL, Oracle, PostgreSQL, Sybase, Informix, and Microsoft SQL Server.

## **6.1. GENERAL USES OF PHP**

- PHP execute system functions, i.e. from files on a system it can create, open, read, write, and close them;
- PHP can manage forms, i.e. collect data from files, save data to a file, through email you can send data, return data to the user;
- You add, delete, and modify elements within your database through PHP;
- Access cookies variables and set cookies;
- Using PHP, you can restrict users to access some pages of your website; and
- It can encrypt data.

### **Characteristics of PHP**

- Simplicity;
- Efficiency;
- Security;
- Flexibility;
- Familiarity;
- Loosely typed language;
- Open source; and
- Object oriented.

## **6.2. HOW TO RUN PHP SCRIPT**

The PHP script or code will execute or run as on web server or a command line interface. PHP codes for the web need to install web server and need

database server. There are various web servers for executing PHP program or code like XAMPP or WAMP and database server like MySQL. Web server like WAMP is supported in windows and XAMPP server is supported both windows and Linux. We learn how to run our PHP program on XAMPP server. Cross platform (X), Apache (A), Maria db (M), PHP (P), Pearl (P) refers to as XAMPP; it is a free distribution server software that makes easy for deploying and testing program on a local web server.

Basic steps for execute PHP program through XAMPP server:

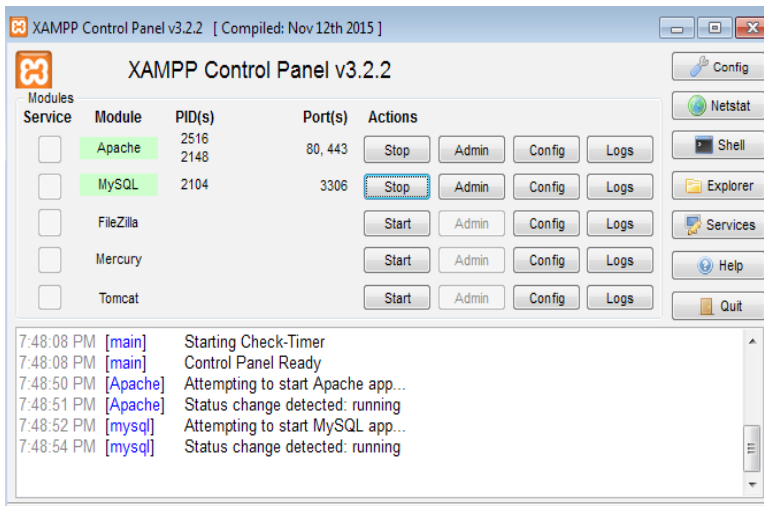
Step 1. Download XAMPP from specific url and install XAMPP in your machine.

Step 2. After successfully installation of XAMPP server, use the XAMPP control panel to start / stop all servers.

Step 3. Write PHP program in a notepad and save it with. php file name.

Step 4. copy. php file inside htdocs folder (C:/Program Files/XAMPP/htdocs).

Step 5. Next start Apache servers and Mysql from the XAMPP control panel and get the dashboard for localhost: search <http://localhost> in any browser.



Step 6. Now run your program, in your browser with <http://localhost/> file. php then it executed.

Step 7. Output will display in your browser.

### 6.3. BASIC SYNTAX OF PHP

A PHP script can be write anywhere in the document. A PHP script starts with `<?php` and ends with `?>`.

Syntax:

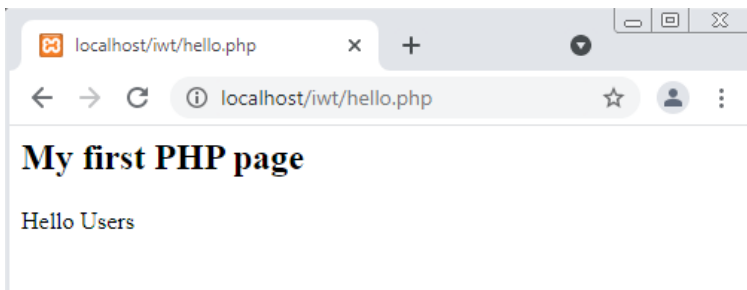
```
<?php
// PHP code goes here
?>
```

By default extension for PHP files is “.php,” in general a PHP file contains HTML tags, and some PHP scripting code. PHP script uses a built-in PHP function “echo” to output the text on a web page. PHP statements end with a semicolon (;).

Example: `hellophp 6.1`

```
<html>
<body>
<h2>My first PHP page</h2>
<?php
echo “Hello Users”;
?>
</body>
</html>
```

Output:



**Figure 6.1.** *Output of `hellophp`.*

1. **PHP Case Sensitivity:** In PHP, keywords (e.g. `if`, `else`, `while`, `echo`, etc.), classes, functions, and user-defined functions are not case-sensitive.



- 2. Comments in PHP:** In PHP code a comment is a line that is not executed as a part of the program, only purpose is to be read by someone who is looking at the code. PHP supports two types of comment:

**i. Single line comment:** As the name propose these are single line or short important details that one can add in there code. To add this, we need to begin the line with (//) or (#).

For Example:

```
<?php
$geek = "hello world!";
echo $geek; // print hello world!
?>
```

**ii. Multi-line or multiple lines comment:** These are used to contain multiple lines with a single tag and can be extended too many lines as required by the user. To add this, we need to begin and end the line with (/\*...\*/).

For Example:

```
<?php
/* This is a multi line comment
In PHP variables are written
by adding a $ sign at the beginning.*/
$geek = "hello world!";
echo $geek;
?>
```

- 3. Creating (Declaring) PHP Variables:** In PHP, name of the variable starts with the \$ sign.

For Example:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

4. **PHP echo and print Statements:** Echo and print are more or less the same. Both are used to output data to the screen. The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. Echo can take multiple parameters while print can take one argument. Echo is slightly faster than print.

## 6.4. PHP CONDITIONAL OR DECISION STATEMENTS

PHP have the following conditional statements:

1. **if statement** – executes some code if a specified condition is true.
  2. **if.else statement** – executes some code if a specified condition is true and another code if that condition is false.
  3. **if.elseif.else statement** – executes different codes for more than two conditions.
  4. **switch statement** – selects one of many blocks of code to be executed.
1. **if statement:** The if statement executes some code if a specified condition is true.

**Syntax:** if (condition) {  
code to be executed if condition is true; }

Example:

For Example:

```
<?php
$t = date("H");
if ($t < "20") {
    echo "Have a good day!";
}
?>
```

**Output:** "Have a good day!" if the current time (HOUR) is less than 20:

2. **if.else statement:** The if. else statement executes some code if a condition is true and another code if that condition is false.

Syntax:

```
if (condition) {
code to be executed if condition is true;
```

```
} else {  
code to be executed if condition is false;  
}
```

For Example:

```
<?php  
$t = date("H");  
if ($t < "20") {  
echo "Have a good day!";  
} else {  
echo "Have a good night!";  
}  
?>
```

**Output:** "Have a good day!" if the current time is less than 20, otherwise "Have a good night!."

- 3. if.elseif.else statement:** The if. elseif. else statement executes different codes for more than two conditions.

Syntax:

```
if (condition) {  
code to be executed if this condition is true;  
} elseif (condition) {  
code to be executed if first condition is false and this condition is true;  
} else {  
code to be executed if all conditions are false;  
}
```

For Example:

```
<?php  
$t = date("H");  
if ($t < "10") {  
echo "Have a good morning!";  
} elseif ($t < "20") {  
echo "Have a good day!";  
} else {
```

```
echo "Have a good night!";  
}  
?>
```

**Output:** “Have a good morning!” if the current time is less than 10, and “Have a good day!” if the current time is less than 20. Otherwise it will output “Have a good night!”:

4. **Switch Statement:** the switch statement to select one of many blocks of code to be executed.

Syntax:

```
switch (n) {  
case label1:  
code to be executed if n=label1;  
break;  
case label2:  
code to be executed if n=label2;  
break;  
case label3:  
code to be executed if n=label3;  
break;  
.  
default:  
code to be executed if n is different from all labels;  
}
```

## 6.5. PHP LOOPS

Loops are used to execute the same block of code again and again, provided that a certain condition is true. In PHP, we have the following loop types:

- **While:** loops through a block of code as long as the specified condition is true
- **Do.while:** loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **For:** loops through a block of code a specified number of times
- **Foreach:** loops through a block of code for each element in an

array

- **While loop:** The while loop executes a block of code as long as the specified condition is true.

Syntax:

```
while (condition) {
code to be executed; // if condition is true
}
```

For Example:

```
<?php
$x = 1;
while($x <= 5) {
echo "The number is: $x <br>";
$x++;
}
?>
```

The example output will display the numbers from 1 to 5:

**Do.while loop:** The do. while loop will always execute the block of code once, then it will check the condition, and repeat the loop while the specified condition is true.

Syntax:

```
do {
code to be executed;
} while (condition is true);
```

**For Example:** <?php

```
$x = 1;
do {
echo "The number is: $x <br>";
$x++;
} while ($x <= 5);
?>
```

Above example first sets a variable  $x = 1$ , then, the do while loop will write some output, and increment the variable  $x$  with 1. After the condition

is checked (is \$x less than, or equal to 5?), and the loop will continue to run as long as \$x is less than, or equal to 5.

- **For loop:** for loop is used when you know how many times the script should be run.

**Syntax:** for (init counter; test counter; increment counter) {  
code to be executed for each iteration;  
}

Factors:

- **init counter:** Initialize the loop counter value.
- **test counter:** Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- **increment counter:** Increases the loop counter value.

For Example:

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

Above example output will display the numbers from 0 to 10:

- **Foreach loop:** foreach loop works only on arrays, and is used to loop throughout each key/value pair in an array.

Syntax:

```
foreach ($array as $value) {
    code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to \$value and the array pointer is moved by one, until it reaches the last array element.

For Example:

```
<?php
$colors = array ("red," "green," "blue," "yellow");
foreach ($colors as $value) {
```

```
echo "$value <br>";  
}  
?>
```

Above example will output the values of the given array (\$colors).

## 6.6. PHP AND HTML, ARRAYS, AND FUNCTIONS

1. **Combine PHP and HTML:** When it comes to using PHP in HTML, there are two different approaches.
  - The first is to embed the PHP code in your HTML file itself with the .html extension; this requires a special consideration; the second way is to combine PHP and HTML tags in .php files. Since PHP is a server-side scripting language, the code is interpreted and run on the server side.

For example, if you add the following code in your index. html file, it won't run out of the box.

```
<html>  
<head>  
<title>Embed PHP in a.html File</title>  
</head>  
<body>  
<h1><?php echo "Hello World" ?></h1>  
</body>  
</html>
```

By default, PHP tags in your .html document are not detected, and they're just considered plain text, outputting without parsing. That's because the server is usually configured to run PHP only for files with the .php extension.

If you want to run your HTML files as PHP, you can tell the server to run your .html files as PHP files, but it's a much better idea to put your mixed PHP and HTML code into a file with the .php extension.

2. **Arrays:** An array is a data structure that stores one or more similar type of values in a single variable. For example if you want to store 10 numbers then in its place of defining 10 variables it's easy to define an array of 10 lengths.

There are three different types of arrays:

- **Indexed arrays** – Arrays with a numeric index
- **Associative arrays** – Arrays with named keys
- **Multidimensional arrays** – Arrays containing one or more arrays
- **Indexed Arrays:** There are two ways to create indexed arrays:

First, index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo," "BMW," "Toyota");
```

or second index can be assigned manually:

```
$cars[0] = "Volvo";
```

```
$cars[1] = "BMW";
```

```
$cars[2] = "Toyota";
```

Example: indexarray 6.2

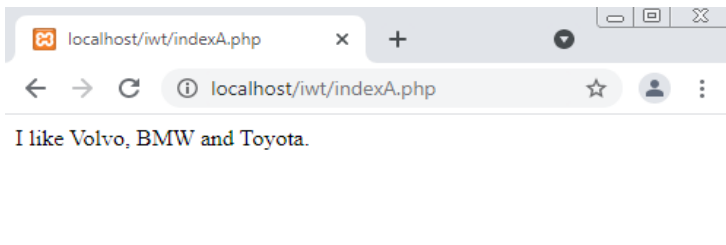
```
<?php
```

```
$cars = array("Volvo," "BMW," "Toyota");
```

```
echo "I like " . $cars[0] . " , " $cars[1] . " and " . $cars[2]. " .";
```

```
?>
```

**Output:**



**Figure 6.2.** *Output of indexarray.*

- **Associative arrays:** Associative arrays are arrays that use named keys that you assign to them. There are two ways to create an associative array:

```
$age = array("Peter"=>"35," "Ben"=>"37," "Joe"=>"43");
```

or:

```
$age['Peter'] = "35";
```

```
$age['Ben'] = "37";
```



```
$age['Joe'] = "43";
```

Example: `associativearray` 6.3

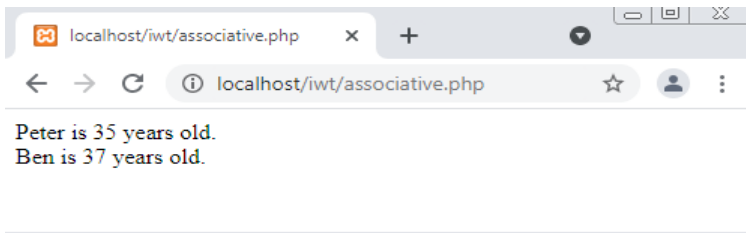
```
<?php
```

```
$age = array("Peter"=>"35,"Ben"=>"37,"Joe"=>"43");
```

```
echo "Peter is " . $age['Peter'] . " years old.";
```

```
?>
```

Output:



**Figure 6.3.** *Output of associativearray.*

- **Multidimensional arrays:** A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple indexes.

Example: `MultisArray` 6.4

```
<html>
```

```
<body>
```

```
<?php
```

```
$cars = array (
```

```
array("Volvo,"22,18),
```

```
array("BMW,"15,13),
```

```
array("Saab,"5,2),
```

```
array("Land Rover,"17,15)
```

```
);
```

```
echo $cars[0][0].": In stock: " . $cars[0][1] . ", sold: " . $cars[0][2] . "<br>";
```

```
echo $cars[1][0].": In stock: " . $cars[1][1] . ", sold: " . $cars[1][2] . "<br>";
```

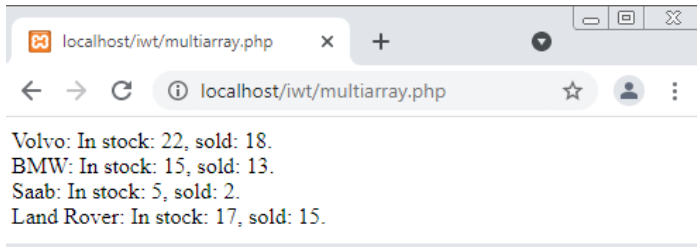
```
echo $cars[2][0].": In stock: " . $cars[2][1] . ", sold: " . $cars[2][2] . "<br>";
```

```
echo $cars[3][0].“: In stock: ”.$cars[3][1]. “, sold: ”.$cars[3][2].“.<br>”;
?>
```

```
</body>
```

```
</html>
```

Output:



**Figure 6.4.** *Output of multiarray.*

3. **Functions:** PHP functions are similarly to other programming languages. A function is a portion of code which takes one more input in the form of parameter and does some process and returns a value. Already have seen many functions like **fopen()** and **fread()** etc. They are built-in functions but PHP gives you option to create your own functions as well.

There are two important parts which should be known:

- Creating a PHP Function
  - Calling a PHP Function
- i. **PHP Function:** when we want to create a function, its name should start with keyword **function** and all the PHP code should be put inside `{and}` braces. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called `writeMessage()` and then calls it just after creating it.

Example: function 6.5

```
<html>
```

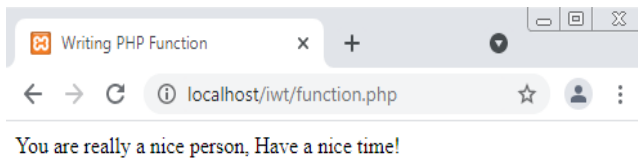
```
<head>
```

```
<title>Writing PHP Function</title>
```

```
</head>
```

```
<body>
<?php
/* Defining a PHP Function */
function writeMessage() {
echo "You are really a nice person, Have a nice time!";
}
/* Calling a PHP Function */
writeMessage();
?>
</body>
</html>
```

Output:



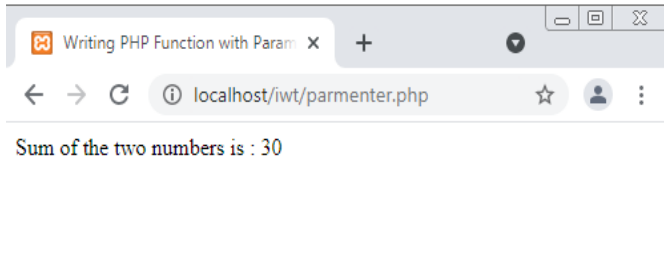
**Figure 6.5.** *Output of function.*

- ii. **PHP functions with parameters:** PHP function provide option to pass your parameters inside a function. You can pass as many as parameters, these parameters work similar to variables inside your function.

Example: parameter 6.6

```
<html>
<head>
<title>Writing PHP Function with Parameters</title>
</head>
<body>
<?php
function addFunction($num1, $num2) {
$sum = $num1 + $num2;
```

```
echo "Sum of the two numbers is: $sum";  
}  
addFunction(10, 20);  
?>  
</body>  
</html>  
Output:
```



**Figure 6.6.** *Output of parameter function.*

## 6.7. PHP STRINGS

A string is a sequence or array of characters, like “welcome you!”.

Property of string functions:

**strlen():** PHP strlen() function returns the length of a string.

For Example:

```
<?php  
echo strlen(“welcome you!”);  
?>
```

Above code output will display 11, the length of the string “welcome you!”:

**str\_word\_count():** PHP function str\_word\_count() counts the number of words in a string.

For Example:

```
<?php  
echo str_word_count(“Hello world!”);  
?>
```

Above code output will display 2, Count the number of word in the string “Hello world!”:

**strrev():** PHP function strrev() return reverses a string.

For Example:

```
<?php
echo strrev(“Hello world!”);
?>
```

Above code output will display !dlrow olleH, Reverse the string “Hello world!”:

**strpos():** PHP function strpos() searches a particular text within a string. If a text is found, the function returns the character position of the first text. If no match is found, it will return FALSE. The first character position in a string is 0 (not 1).

For Example: Search the text “world” in the string “Hello world!”:

```
<?php
echo strpos(“Hello world!,” “world”);
?>
```

Above code output will display 6.

**str\_replace():**PHP function str\_replace() replaces a number of characters with some other characters in a string.

For Example:

```
<?php
echo str_replace(“world,” “Lolly,” “Hello world!”);
?>
```

Above code output will display Hello Lolly!, Replace the text “world” with “Lolly”:

**String Concatenation:** To concatenate two string variables together, use the dot (.) operator –

For Example:

```
<?php
$string1=“Hello World”;
$string2=“12345”;
echo $string1. “.” $string2;
```

?>

Above code output will display Hello World 12345.

## 6.8. FORM PROCESSING AND FILES

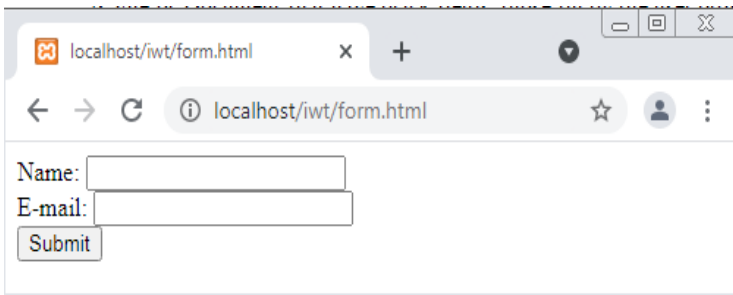
A file or document that have black fields, those fill by the user provided data or user can select the data. In an informal way the data will store in the data base. PHP have two main super globals `$_GET` and `$_POST` are used to collect form-data.

**A simple HTML form:** a simple HTML form with two input fields and a submit button will display by the under code:

Example: `htmlform 6.7`

```
<html>
<body>
<form action="welcome. php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

Output:



**Figure 6.7.** Output of `htmlform`.

The above form fills out by the user and clicks the submit button, the form data is sent with the HTTP POST method for processing to a PHP file named "welcome .php." For display the submitted data you could simply echo all the variables in "welcome .php" code.

Example: welcome.php 6.8

```
<html>
```

```
<body>
```

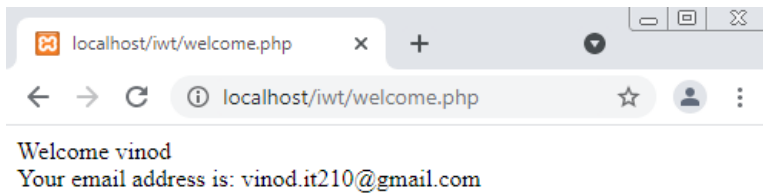
```
Welcome <?php echo $_POST["name"]; ?><br>
```

```
Your email address is: <?php echo $_POST["email"]; ?>
```

```
</body>
```

```
</html>
```

Output:



**Figure 6.8.** *Output of welcome.php.*

Note: The same outcome could also be achieved using the HTTP GET method.

Example:

```
<html>
```

```
<body>
```

```
<form action="welcome_get.php" method="get">
```

```
Name: <input type="text" name="name"><br>
```

```
E-mail: <input type="text" name="email"><br>
```

```
<input type="submit">
```

```
</form>
```

```
</body>
```

```
</html>
```

and "welcome\_get.php" looks like this:

```
<html>
```

```
<body>
```

```
Welcome <?php echo $_GET["name"]; ?><br>
```

```
Your email address is: <?php echo $_GET["email"]; ?>  
</body>  
</html>
```

**When to use GET method:** the user provided data or Information sent from a form with the GET method, which sent data is visible to everyone (all variable names and values are displayed in the URL). GET method also has limits about 2000 characters on the amount of information to be send. Because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases but GET should not be used for sending passwords or other sensitive information.

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY\_STRING environment variable.
- The PHP provides **\$\_GET** associative array to access all the sent information using GET method.

**When to use POST method:** the user provided Data or Information sent from a form with the POST method is invisible to others (all names/values are embedded within the HTTP request) and has no limits on the amount of information to send. POST method supports advanced functionality such as multi-part binary input while uploading files to server.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **\$\_POST** associative array to access all the sent information using POST method.



### 6.8.1. Files Manipulation

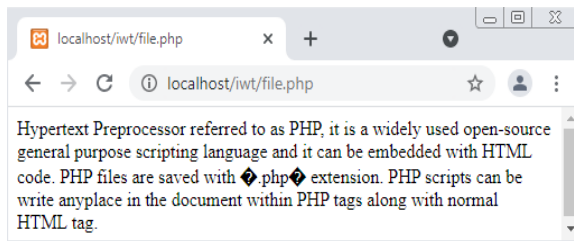
PHP has several file manipulation functions for create, read, upload, and edit files.

- **Open file: fopen():** In fopen() function first parameter contains the name of the file to be opened and the second parameter indicate in which mode the file should be opened.

Example: fopen 6.9

```
<?php
$myfile = fopen("webdictionary.txt","r") or die("Unable to open file!");
echo fread($myfile, filesize("webdictionary.txt"));
fclose($myfile);
?>
```

Output:



**Figure 6.9.** Output of fopen.

The file could be opened in one of the following modes:

Modes	Description
r	<b>Open a file for read only.</b> File pointer starts at the beginning of the file
w	<b>Open a file for write only.</b> Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	<b>Open a file for write only.</b> The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	<b>Creates a new file for write only.</b> Returns FALSE and an error if file already exists

r+	<b>Open a file for read/write.</b> File pointer starts at the beginning of the file
w+	<b>Open a file for read/write.</b> Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	<b>Open a file for read/write.</b> The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	<b>Creates a new file for read/write.</b> Returns FALSE and an error if file already exists

- **Read file: fread():** The fread() function reads from an open file, first parameter of fread() contains the name of the file to read from and the second parameter indicate the maximum number of bytes to read. The following PHP code reads the “webdictionary.txt” file to the end.

```
fread($myfile, filesize("webdictionary.txt"));
```

- **Close file: fclose():** The fclose() function is used to close an open file. The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

```
<?php
```

```
$myfile = fopen("webdictionary.txt," "r");
```

```
// some code to be executed.
```

```
fclose($myfile);
```

```
?>
```

- **Write to file: fwrite():** The fwrite() function is used to write to a file, in first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

Example: writefile 6.10

```
<?php
```

```
$myfile = fopen("newfile.txt," "w") or die("Unable to open file!");
```

```
$txt = "Vinod Kumar\n";
```

```
fwrite($myfile, $txt);
```

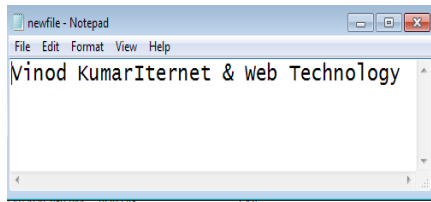
```
$txt = "Internet & Web Technology\n";
```

```
fwrite($myfile, $txt);
```

```
fclose($myfile);
```

?>

Output: after run writefile 6.10 code, result will display in newfile.text



**Figure 6.10.** *Output of writefile.*

## 6.9. COOKIES AND SESSIONS

### 6.9.1. Cookies

Cookies are a text files that stored on the client site computer and they are reserved for tracking purpose. PHP clearly supports HTTP cookies, there are three steps concerned in identifying and returning users:

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
  - Browser stores this information on local machine for future use.
  - When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.
1. **Create Cookies:** A cookie is created with the `setcookie()` function, Only the name parameter is required other parameters are optional.

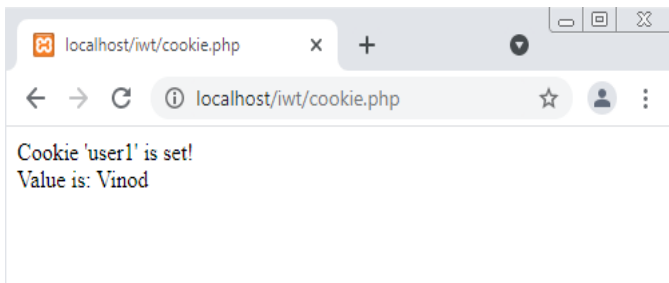
Syntax:

`setcookie (name, value, expire, path, domain, secure, httponly);`

2. **Create/Retrieve a Cookie:** in below example creates a cookie named “user1” with the value “Vinod.” The cookie will expire after 30 days ( $86400 * 30$ ), “/” means that the cookie is available in entire website (otherwise, select the directory you prefer). We then retrieve the value of the cookie “user” (using the global variable `$_COOKIE`). We also use the `isset()` function to find out if the cookie is set. Remember that `setcookie()` function should appear before the `<html>` tag.

**Example: createcookie 6.11**

```
<?php
$cookie_name = "user1";
$cookie_value = "Vinod";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); //
86400 = 1 day
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name]))
{
    echo "Cookie named '" . $cookie_name . "' is not set!";
}
else
{
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

**Output:****Figure 6.11.** *Output of createcookie 6.11.*

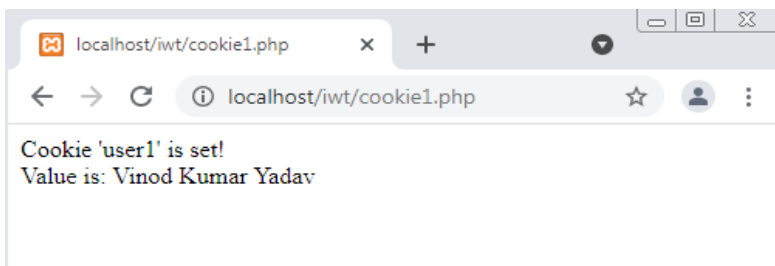
- 3. Modify a cookie value:** To modify a cookie, set again the cookie value using the `setcookie()` function:

**Example: ModifyCookie 6.12**

```
<?php
$cookie_name = "user1";
$cookie_value = "Vinod Kumar Yadav";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>

<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

Output:



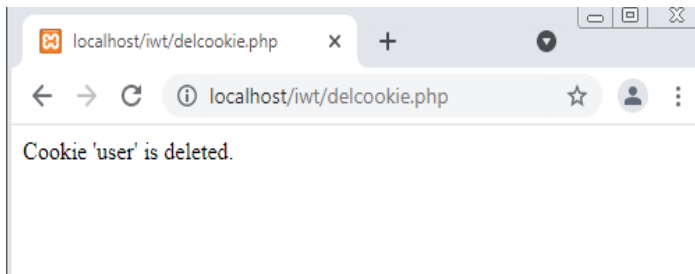
**Figure 6.12.** *Output of ModifyCookie 6.12.*

- 4. Delete a cookie:** To delete a cookie, use the `setcookie()` function with an ending date in the past:

Example:DeleteCookie 6.13

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body>
</html>
```

Output:



**Figure 6.13.** *Output of DeleteCookie 6.13.*

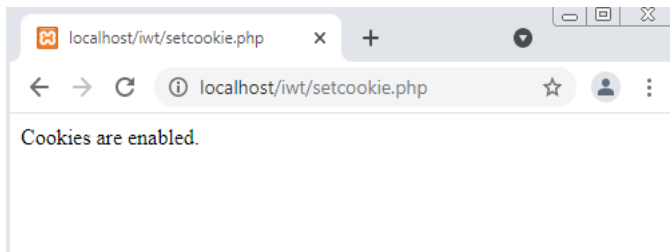
- 5. Check if cookies are enabled:** to check whether cookies are enabled, try to create a test cookie with the `setcookie()` function, then count the `$_COOKIE` array variable.

#### **Example: enabled 6.14**

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>
<?php
if(count($_COOKIE) > 0) {
echo "Cookies are enabled.";
```

```
} else {  
echo "Cookies are disabled.";  
}  
?>  
</body>  
</html>
```

### Output:



**Figure 6.14.** *Output of enabled 6.14.*

## 6.9.2. PHP Session

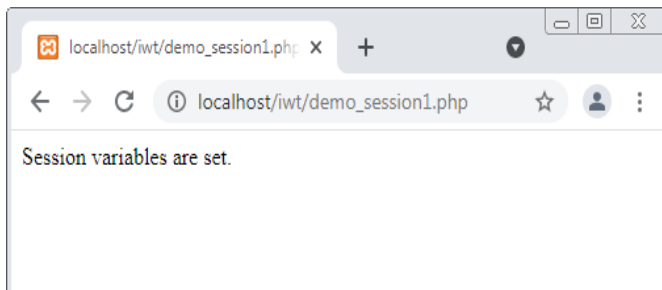
At what time work with an application, you open it, perform some modification, and then you close it. This is a great deal similar to a Session. The computer application knows who you are, when you start and when you end the application. But on the online (internet) there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't keep status of users.

With the help of Php Session variables resolve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc.). By default, session variables end until the user closes the browser so Session variables hold information about one single user, and are available to all pages in one application. If you need a permanent storage, you may want to store the data in a database.

- 1. Start a PHP session:** A session is started with the `session_start()` function. Session variables are set with the PHP global variable: `$_SESSION`.

Example: `demo_session1.php` 6.15

```
<?php
// Start the session
session_start();
?>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>
```

**Output:**

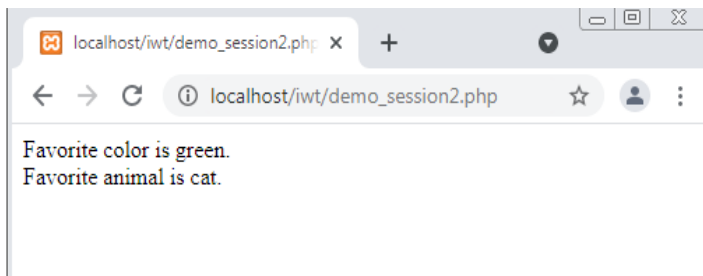
**Figure 6.15.** *Output of demo\_session1.php 6.15.*

- 2. Get PHP session variable values:** Next, we create another page called "demo\_session2. php," from this page, we will access the session information we set on the first page ("demo\_session1. php"). Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session\_start()). Also all session variable values are stored in the global \$\_SESSION variable:



**Example: demo\_session2.php 6.16**

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . "<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

**Output:**

**Figure 6.16.** *Output of demo\_session2.php 6.16.*

- 3. Modify a PHP session variable:** To modify a session variable, now overwrite code:

```
<?php
session_start();
?>
<html>
<body>
```

```
<?php
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
</body>
</html>
```

- 4. Destroy a PHP session:** To remove all global session variables and destroy the session, use `session_unset()` and `session_destroy()`:

```
<?php
session_start();
?>
<html>
<body>
<?php
session_unset();
session_destroy();
?>
</body>
</html>
```

## 6.10. BASIC OBJECT ORIENTED PROGRAMING IN PHP

OOP referred to as Object-oriented programing (OOP). Procedural programing is concern to writing procedures or functions that perform operations on the data, while OOP is about creating objects that contain both data and functions.

Object-oriented programing has various advantages:

- OOP is easier and faster to execute
- OOP provides a clear structure for the programs
- OOP helps to maintain the PHP code easier to continue, modify and debug
- OOP makes to create full reusable applications with less code and development time

### 6.10.1. Classes and Objects

A class is a template for objects, and an object is an instance of class.

**Define a Class:** A class in php is defining by using the class keyword, followed by the class name and a pair of curly braces ({}). In class variables are called properties and functions are called methods all define inside the braces.

**Syntax:** <?php  
 class classname {  
 // code goes here.  
 }  
 ?>

Below example we declare a class named student consisting of two properties (\$enroll and \$name) and two methods set\_name() and get\_name() for setting and getting the \$name property:

Example:

```
<?php
class student
{
    public $enroll; // Properties
    public $name;
    function set_name($name) // Methods
    {
        $this->name = $name;
    }
    function get_name() {
        return $this->name;
    }
}
?>
```

**Define Objects:** An object of a class is created using the new keyword. Without objects classes are nothing, we can create multiple objects from a class. Each object has all the properties and methods defined inside the class, but they will have different property values.

**\$this Keyword:** the \$this keyword refers to the current object, and is only available inside methods.

For Example:

```
<?php
class student {
public $name;
public $enroll;
function set_name($name) {
$this->name = $name;
}
function get_name() {
return $this->name;
}
function set_enroll($enroll) {
$this->enroll = $enroll;
}
function get_enroll() {
return $this->enroll;
}
}
$shubh = new student();
$shubh->set_name('shubh');
$shubh->set_enroll('111');
echo "enroll: " . $shubh->get_enroll();
echo "<br>";
echo $shubh->get_name();
echo "<br>";
?>
```

We can change the value of the \$name property, There are two ways:

1. Inside the class use \$this:

```
<?php
```

```
class Fruit {  
    public $name;  
    function set_name($name) {  
        $this->name = $name;  
    }  
}  
  
$apple = new Fruit();  
$apple->set_name("Apple");  
?>
```

2. outside the class by directly changing the property value:

```
<?php  
class Fruit {  
    public $name;  
}  
  
$apple = new Fruit();  
$apple->name = "Apple";  
?>
```

**instanceof:** the instance of keyword to check if an object belongs to a specific class:

```
<?php  
$apple = new Fruit();  
var_dump($apple instanceof Fruit);  
?>
```



## CHAPTER 7

# PHP WITH MYSQL

### CONTENTS

7.1. Basic of MySQL.....	200
7.2. Create Table and Insert Data Using PHP .....	204
7.3. Altering Tables Using PHP.....	208
7.4. Update Data in a MYSQL Table .....	211
7.5. Delete Data and Tables From Database.....	213
7.6. PHP My Admin Features .....	217

Through PHP script, can connect, insert data and manipulate databases. MySQL is the most popular database connection system used with PHP code.

## 7.1. BASIC OF MYSQL

- MySQL is a standard form of the SQL data language.
- MySQL is a database system that used to store data collected from the PHP web page.
- MySQL is a database system that runs on a different web server.
- MySQL is perfect for both small and large applications database.
- MySQL is very fast, reliable, consistence and easy to use.
- MySQL employ standard of SQL commands.
- MySQL supports large databases, up to million rows or more in a table.
- MySQL compiles and run on several platforms.
- MySQL is free to download and utilize.
- MySQL is developed, distributed and supported by Oracle Corporation

The data or information is stored in the form of tables in MySQL database. A table is a collection or representation of related data in rows and columns form. Database is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc. For Example, university database organizes the data about students, faculty, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.

### 7.1.1. Administrative MySQL Command

Some important MySQL commands, which you will work with MySQL database:

- **Database name:** this will be used to select a database in the phpMyAdmin panel.
- **View databases:** list out the databases that are accessible or created by the MySQL DBMS.
- **View tables:** Shows the tables in the database once a database has been selected with the use command.



- **Display columns from tablename:** Shows the attributes, types of attributes, key information, whether NULL is permitted, defaults, and other information for a table.
- **Display index from tablename:** Presents the details of all indexes on the table, including the PRIMARY KEY.
- **Display table status like tablename:** Reports details of the MySQL DBMS performance and statistics.

### 7.1.2. Basic Steps for PHP Script to Connect with MySQL

Step 1. Download XAMPP from specific url and install XAMPP in your machine.

Step 2. After successfully installation of XAMPP server, use the XAMPP control panel

Step 3. Next start Apache servers and Mysql from the XAMPP control panel



**Figure 7.1.** XAMPP control panel.

Step 4. Write PHP program (follow step 4.1 to 4.3) in a notepad and save it with. php file name.

Step 4.1 Open a Connection to MySQL: We can access data in the MySQL database; we require connection with the server by use the following code.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
```

```

$conn = new mysqli($servername, $username, $password); // Create
connection
if ($conn->connect_error) // Check connection
{
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>

```

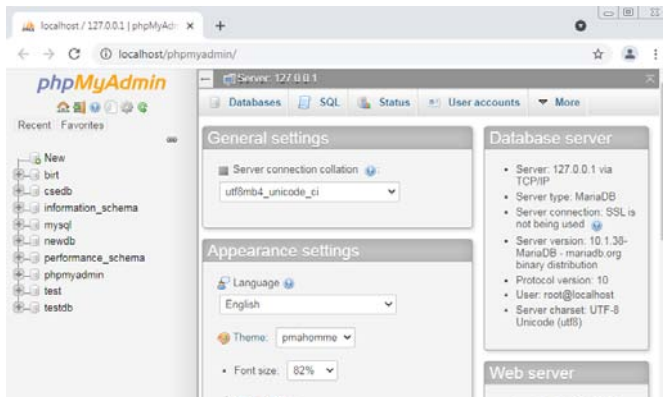
Step 4.2 Creating Database: If the connection is successful, write a SQL query to create a database and store it in a string variable and execute query.

```

$sql = "CREATE DATABASE newDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully with the name newDB";
} else
{
    echo "Error creating database: " . $conn->error; }
OR

```

We also create database by use XAMPP control panel from Mysql Admin with phpMyAdmin panel.



**Figure 7.2.** *PhpMyAdmin window.*

Step 4.3: Close the Connection: In MySQL create connection will be closed automatically when the script ends. To close the connections use the following code.

```
$conn->close();
```

Step 5. Copy. php file inside htdocs folder (C:/Program Files/XAMPP/htdocs).

Step 6. Now run your program, in your browser with http://localhost/file.php then it executed.

Step 7. Output will display in your browser.

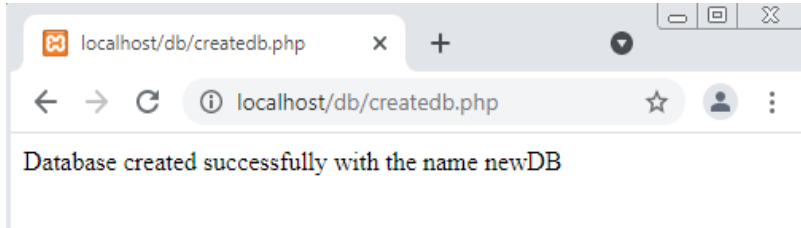
### 7.1.3. Create Database Using PHP

Specify the three arguments servername, username and password to the mysqli object whenever creating a database.

Example: createdatabase.php 7.1

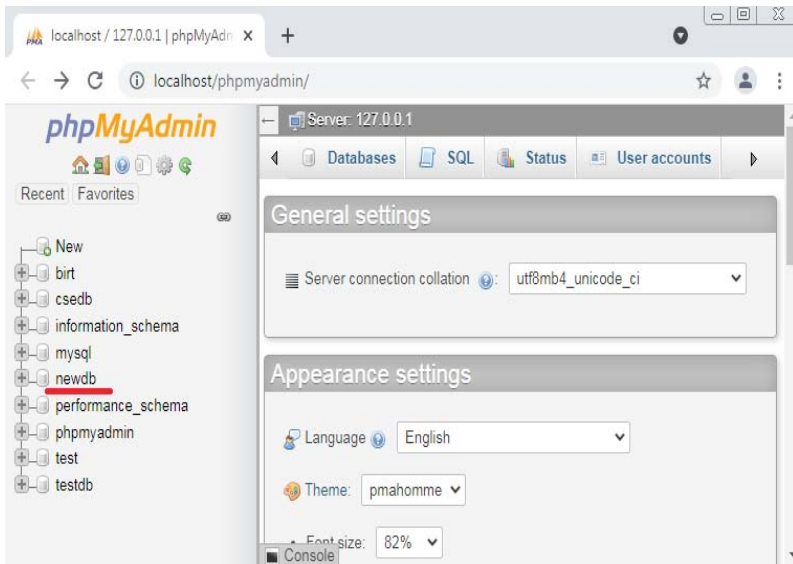
```
<?php
$servername = "localhost";
$user='root';
$pass="";
$conn = new mysqli($servername, $user, $pass); // Creating a connection
if ($conn->connect_error) // Check connection
{
    die("Connection failed: ". $conn->connect_error);
}
$sql = "CREATE DATABASE newDB"; // Creating a database named
newDB
if ($conn->query($sql) === TRUE)
{
    echo "Database created successfully with the name newDB";
} else
{
    echo "Error creating database: ". $conn->error;
}
$conn->close(); // closing connection
?>
```

Output:



**Figure 7.3.** *createdatabase.php*

After successfully run `createdatabase.php` file, a fresh `newDB` database created that will display in the phpMyAdmin panel as shown in Figure 7.4.



**Figure 7.4.** *Newdb database in phpMyAdmin.*

## 7.2. CREATE TABLE AND INSERT DATA USING PHP

### 7.2.1. Create Table Using PHP

A database has table its own unique name and consists of columns and rows. The `CREATE TABLE` statement is used to create a table in MySQL. We will create a table named “Guests,” with five columns: “id,” “firstname,” “lastname,” “email” and “reg\_date.”

## Command for create table:

```
CREATE TABLE Guests (id INT(6) UNSIGNED AUTO_INCREMENT
PRIMARY KEY, firstname VARCHAR(30) NOT NULL, lastname
VARCHAR(30) NOT NULL, email VARCHAR(50), reg_date TIMESTAMP
DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_
TIMESTAMP)
```

- Notes on the above command:

The data type specifies which type of data can hold by the column. After the data type, you can identify other optional attributes for each column:

- **Not Null:** Each row must contain a value for that column, null values are not allowed.
- **Default value:** Set a default value that is added when no other value is passed.
- **Unsigned:** Used for number types, limits the stored data to positive numbers and zero.
- **Auto increment:** MySQL automatically increases the value of the field by 1 each time a new record is added.
- **Primary key:** Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO\_INCREMENT.

Below given examples show how to create the table with PHP:

Example: createtable.php 7.2

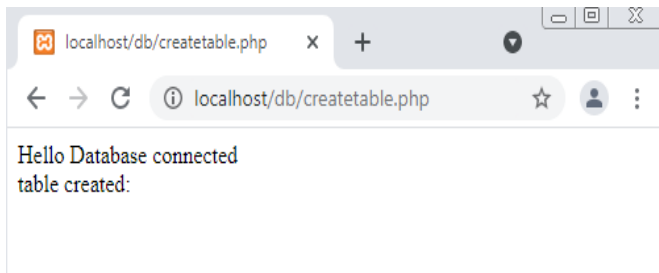
```
<?php
$user='root';
$password='';
$db='newdb';
$db=new mysqli('localhost',$user,$password,$db) or die ("unable to
connect");
echo "Hello Database connected";
echo "<br>";
$sql="create table Student (id INT(6),Fname varchar(20), Lname
varchar(20),contact INT(6))";
if($db->query($sql)==TRUE)
```

```

{
echo“table created.”;
}
else
{
echo“error.”$db->error;
}
$db->close();
?>

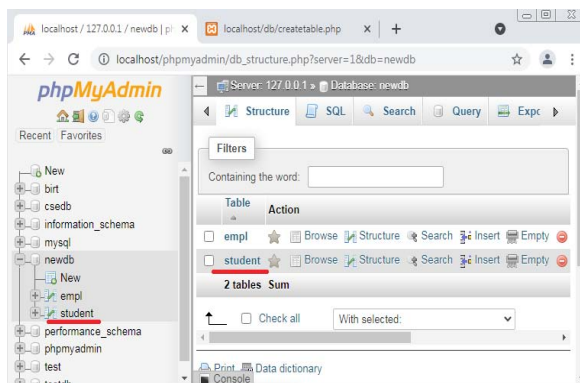
```

Output:



**Figure 7.5.** *Createtable.php.*

After successfully run createtable.php file, a table Student created inside the newDB database that will display in the phpMyAdmin panel as shown in figure 7.6.



**Figure 7.6.** *Student table inside the newdb database.*

### 7.2.2. Insert Data Using PHP

Once a database and a table have been created, we can start adding data in created table. Here are some rules to follow:

- The SQL query must be quoted in PHP;
- String values inside the SQL query must be quoted;
- Numeric values must not be quoted; and
- The word NULL must not be quoted.

**This command is used to add new records to a MySQL table:**

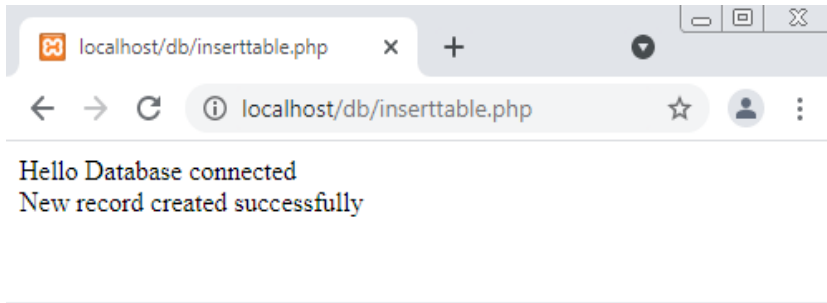
INSERT INTO table\_name (column1, column2, column3.) VALUES (value1, value2, value3.)

Below given examples show how to add a new record to the “student” table.

Example: insertdata.php 7.5

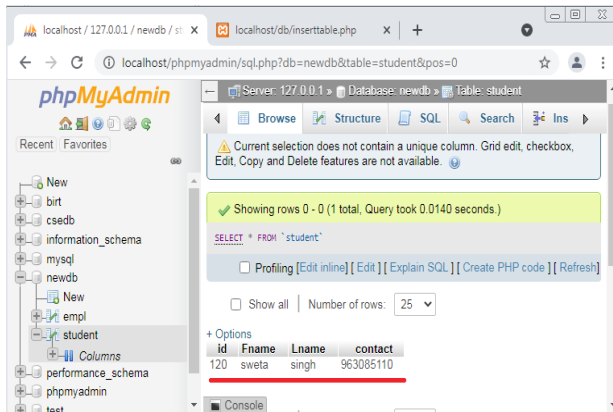
```
<?php
$user='root';
$password='';
$db='newdb';
$db=new mysqli('localhost',$user,$password,$db) or die ("unable to
connect");
echo "Hello Database connected <br>";
$sql="insert into Student (id, Fname, Lname, contact)
values (120,'sweta','singh','963085110)";
if($db->query($sql)==TRUE)
{
echo"New record created successfully";
}
else
{
echo"error."$db->error;
}
$db->close();
?>
```

Output:



**Figure 7.7.** *insertdata.php.*

After successfully run insertdata.php file, inside table Student some values inserted in a new row that all belongs to the newDB database that will display in the phpMyAdmin panel as shown in Figure 7.8.



**Figure 7.8.** *Insert values inside student table.*

## 7.3. ALTERING TABLES USING PHP

We can use ALTER (Structure of Tables) to add and delete the properties of a column in a mysql database using php.

- 1 Adding A column (ADD)
- 2 Deleting A column (DROP)
1. **Adding a column:** The following code adds a new column to the table Student that already created:



**This command is used to add new records to a MySQL table:**

```
$sql="ALTER TABLE Student ADD city varchar(20)";
```

Below given examples show how to add a new column to the “student” table.

Example: alter\_add.php 7.6

```
<?php
```

```
$user='root';
```

```
$pass="";
```

```
$db='newdb';
```

```
$db=new mysqli('localhost',$user,$pass,$db) or die ("unable to connect");
```

```
echo "Hello Database connected <br>";
```

```
$sql="alter table Student ADD city varchar(20)";
```

```
if($db->query($sql)==TRUE)
```

```
{
```

```
echo"New column add successfully";
```

```
}
```

```
else
```

```
{
```

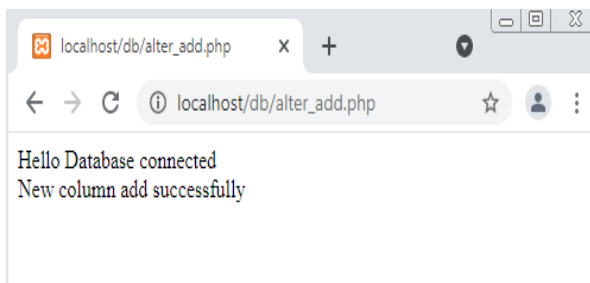
```
echo"error."$db->error;
```

```
}
```

```
$db->close();
```

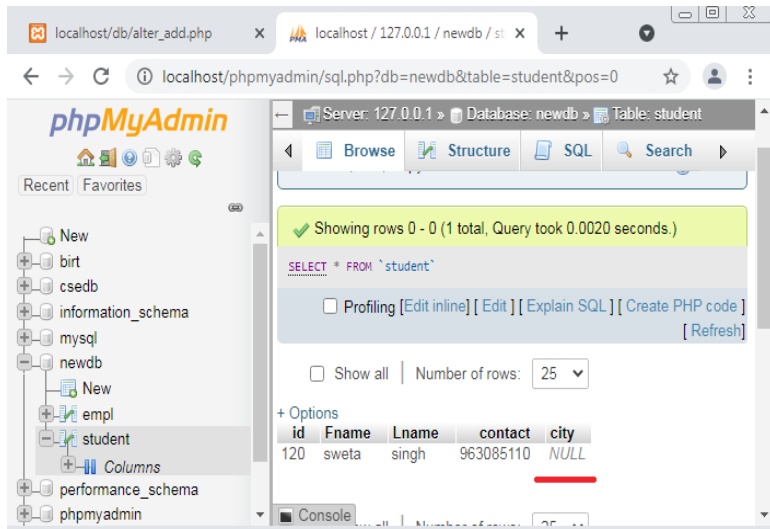
```
?>
```

**Output:**



**Figure 7.9.** Add new column inside student table.

After successfully run `alter_add.php` file, inside table `Student` inserted a new column 'city' that all belongs to the newDB database that will display in the phpMyAdmin panel as shown in figure 7.10.



**Figure 7.10.** New column *city* inside *student* table.

2. **Delete a column:** This code deletes all field or column of a given name from the table.

**This command is used to drop a Column to MySQL table:**

```
$sql="ALTER TABLE Student DROP city";
```

Below given example show how to delete 'city' column to the "student" table.

Example: `alter_drop 7.7`

```
<?php
```

```
$user='root';
```

```
$pass="";
```

```
$db='newdb';
```

```
$db=new mysqli('localhost',$user,$pass,$db) or die ("unable to connect");
```

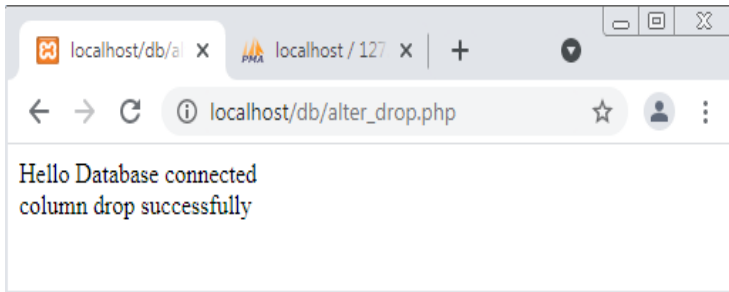
```
echo "Hello Database connected <br>";
```

```
$sql="alter table Student DROP city";
```

```
if($db->query($sql)==TRUE)
```

```
{  
echo"column drop successfully";  
}  
else  
{  
echo"error."$db->error;  
}  
$db->close();  
?>
```

Output:



**Figure 7.11.** *Drop column inside student table.*

After successfully run `alter_drop.php` file, inside student table city column deleted again display as the previous Figure 7.8.

## 7.4. UPDATE DATA IN A MYSQL TABLE

The UPDATE statement is used to update existing records in a table:

Syntax:

UPDATE table\_name SET column1=value, column2=value2. WHERE some\_column=some\_value

The WHERE clause specifies which record or records that should be updated. If you skip the WHERE clause, all records will be updated.

Consider a “Guests” table that have some values like:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

If we want to update the record where id=2 in the “Guests” table then command like this:

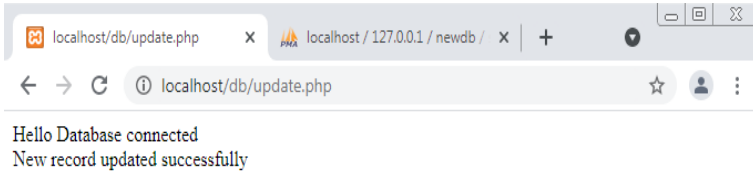
```
UPDATE Guests SET firstname='Sweta' WHERE id=2;
```

Below given example show how to update record to the “student” table.

### **Example: update.php 7.8**

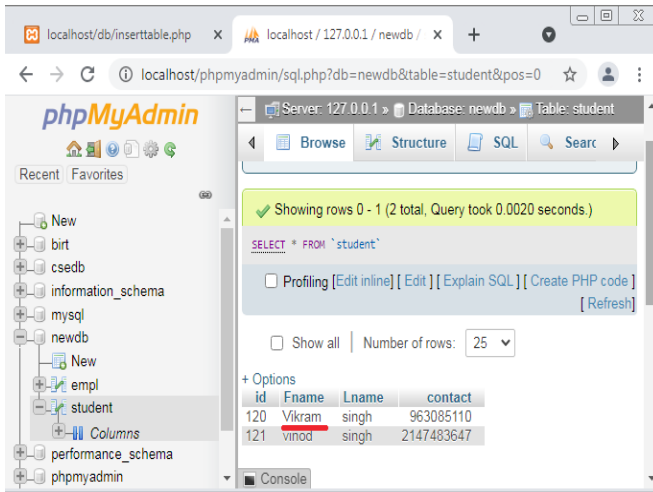
```
<?php
$user='root';
$password='';
$db='newdb';
$db=new mysqli('localhost',$user,$password,$db) or die ("unable to
connect");
echo "Hello Database connected <br>";
$sql="update Student set Fname='Vikram' where id=120";
if($db->query($sql)==TRUE)
{
    echo"New record updated successfully";
}
else
{
    echo"error."$db->error;
}
$db->close();
?>
```

### **Output:**



**Figure 7.12.** *Update row inside student table.*

After successfully run update.php file, inside Student table a row updated that all belongs to the newDB database that will display in the phpMyAdmin panel as shown in Figure 7.13.



**Figure 7.13.** *Update Sweta to Vikram inside student table.*

## 7.5. DELETE DATA AND TABLES FROM DATABASE

- **PHP MySQL Delete Data:** The DELETE statement is used to delete records from a table:

Syntax:

DELETE FROM table\_name WHERE some\_column = some\_value;

The WHERE clause specifies which record or records that should be deleted. If you skip the WHERE clause, all records will be deleted.

Consider a table “Guests”:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

If we want to delete the record where id=2 in the “Guests” table then command like this:

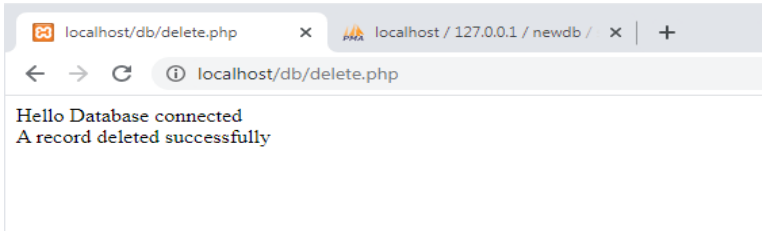
```
DELETE FROM Guests WHERE id = 2;
```

Below given example show how to delete record to the “student” table.

**Example: delete.php 7.9**

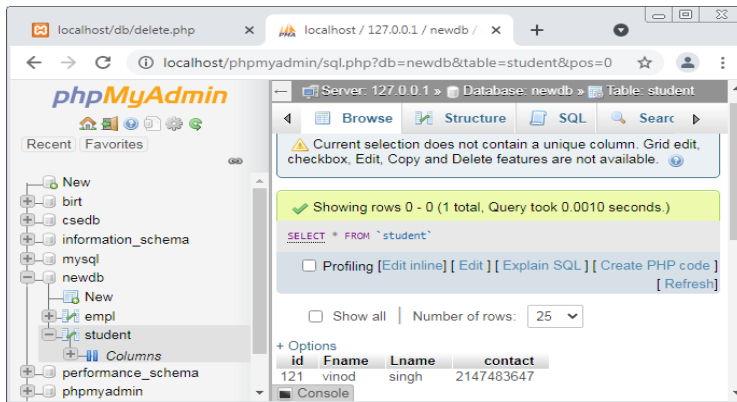
```
<?php
$user='root';
$password='';
$db='newdb';
$db=new mysqli('localhost',$user,$password,$db) or die ("unable to
connect");
echo "Hello Database connected <br>";
$sql="delete from Student where id=120";
if($db->query($sql)==TRUE)
{
    echo "New record deleted successfully";
}
else
{
    echo"error."$db->error;
}
$db->close();
?>
```

Output:



**Figure 7.14.** Deleted record id=120 inside Student table.

After successfully run delete .php file, inside Student table a row deleted that all belongs to the newDB database that will display in the phpMyAdmin panel as shown in Figure 7.15.



**Figure 7.15.** Student table in newdb database.

Delete all the column data from a table:

Syntax:

`DELETE * FROM table_name;`

**Example:** delete \* from student;

**Note:** This command will delete all the column values inside the student table but not delete the table structures.

**Delete table:**

Syntax:

`DELETE FROM table_name;`

**Example:** delete from student;

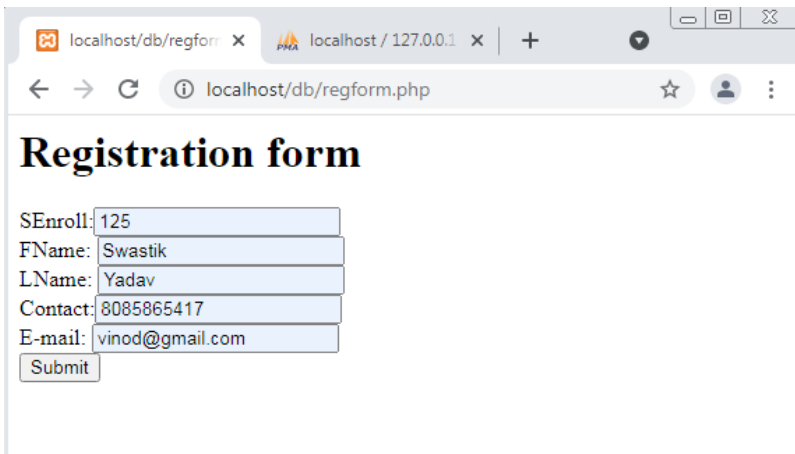
**Note:** This command will delete the student table.

**An example given below to collect data with help of php file and store in database:**

**regform.php:**

```
<html>
<body>
<form action="insertform.php" method="post">
<h1> Registration form </h1>
SEnroll:<input type="int" name="id"><br>
FName: <input type="text" name="fname"><br>
LName: <input type="text" name="lname"><br>
Contact:<input type="int" name="contact"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

Output:



The screenshot shows a web browser window with the address bar displaying 'localhost/db/regform.php'. The page title is 'Registration form'. The form contains the following fields and values:

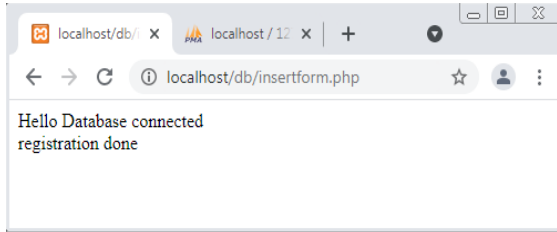
Field	Value
SEnroll:	125
FName:	Swastik
LName:	Yadav
Contact:	8085865417
E-mail:	vinod@gmail.com

Below the fields is a 'Submit' button.

**Figure 7.16.** *Registration form data insert into student table.*

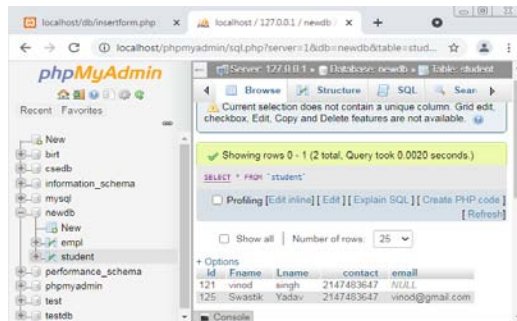


After successfully Submit button will display as 7.17:



**Figure 7.17.** *After submit button.*

Inside Student table a row inserted that submitted by registration form all belongs to the newDB database that will display in the phpMyAdmin panel as shown in Figure 7.18.



**Figure 7.18.** *Inside Student table inserted data.*

## 7.6. PHP MY ADMIN FEATURES

The most familiar and require applications for MySQL management is the phpMyAdmin panel. phpMyAdmin is an administration tool for MySQL database which is open source and free. It is greatly used in managing the database for websites which have been designed using Joomla, WordPress and other platforms for content management. phpMyAdmin is used by many web hosts who provide convenient database organizing services to their customers.

The major phpMyAdmin features are:

- Intuitive web interface.
- Support for most MySQL features:
- browse and drop databases, tables, views, fields, and indexes.

- create, copy, drop, rename and alter databases, tables, fields, and indexes.
- maintenance server, databases, and tables, with proposals on server configuration.
- execute, edit and bookmark any SQL-statement, even batch-queries.
- manage stored procedures and triggers.
- Import data from CSV and SQL.
- Export data to various formats: CSV, SQL, XML, PDF, ISO/IEC 26300, OpenDocument text and spreadsheet, Word, LATEX, and others.
- Creating complex queries using query-by-example (QBE).
- Searching globally in a database or a subset of it.
- Transforming stored data into any format using a set of predefined functions, like displaying BLOB-data as image or download-link.

## CHAPTER 8

# INTRODUCTION WEB TO PYTHON

## CONTENTS

8.1. Advantages of Developing Web Applications in Python .....	220
8.2. Introduction to Python .....	221
8.3. Python with HTML .....	224
8.4. Submitting Forms with Python .....	226
8.5. Providing Text Area with Python .....	229
8.6. Checking Boxes with Python .....	230
8.7. Radio Buttons with Python.....	233
8.8. Drop Down (Select) Option with Python .....	235
8.9. Upload Files with Python.....	237

Web development became famous due to the common accessibility of the web, especially for business or commercial purposes they could offer their products and services on the web. These web services produced a demand for web development that has never slow down. To become a skilled web developer, you need to be familiar with the foundation principles that the web is built with, such as HTTP requests and responses, the client (normally web browsers) and server (web servers such as XAMPP and Apache architectures), HTML, CSS, JavaScript, XML and PHP among many other resources provide a range of perspective and when combined together. It is also important to understand that the core concepts of HTTP, URLs, and HTML.

Web development can be divided into three most important parts:

- Backend development (design business logic, data storage, and processing).
- Frontend development (User interface design).
- API/middleware development (backend and frontend apps communicate).

Python web script can be used in server-side web applications. Python web script is not used in a web browser. The web script executed inside browsers such as Chrome, Firefox and Internet Explorer, Python web script is executed on the server side while JavaScript is downloaded to the client and run by the web browser.

## 8.1. ADVANTAGES OF DEVELOPING WEB APPLICATIONS IN PYTHON

- **Simple to learn:** Python programming language is the most popular language for first-time learners or beginner. The Python language has common expressions and whitespace, which allows you to write considerably less code compared to some other languages like C++ or Java. Not only that, but it has a lesser difficulty of entry because it's comparatively more similar to your daily basis language so you can easily understand and learn the code.
- **Rich environment and libraries:** Python programming language propose a huge range of library tools and packages, which allow you to access a lot of pre-define code, reorganization your application development time. For example, you have to use Numpy and Pandas for mathematical analysis, use Pygal for

charting, and use SQLAlchemy for database queries. Python language also offers wonderful web frameworks like Django and Flask.

- **Faster:** As python programing takes considerably less time to create your projects compared to other programing languages, your thoughts come to life a lot faster, allowing you to gain response and iterate speedily. This fast development time makes Python language particularly great for startups who can hit the market faster to gain a competitive edge.
- **Extensive popularity:** Python programing language is one of the most popular languages in the world, with communities from all over the world. Python language is continuously updated with new features and libraries, at the same time providing excellent documentation and community support.

Particularly for new developers, Python provides extensive support and framework for one to start on their programing journey.

## 8.2. INTRODUCTION TO PYTHON

Python programing language is a high-level programing language that is developed by the Python “interpreter” to generate results. The Python programing language was introduced or developed by Mr. Van Rossum in the between eighties and nineties at the National Research institute for Mathematics and Computer Science in the Netherlands. Python is resultant from many other programing languages including C, C++. Python is proposed to be highly readable it uses English keyword commonly where other programing languages may use punctuation.

Python’s key features that make it an attractive choice of language for beginner:

- **Python is free:** it is open source distribute software
- **Python is easy to learn:** it has a simple language syntax
- **Python is easy to read:** it is uncluttered by punctuation
- **Python is easy to maintain:** it is modular for simplicity
- **Python is interactive:** it has terminal for debugging and testing snippets of code
- **Python is portable:** run on wide variety of hardware platforms and has the same interface on all platforms

- **Python use interpreter:** there is no need to compilation required for python code
- **Python is high-level programing language:** it uses English keyword commonly and has automatic memory management
- **Python is Versatile:** supports both procedure-oriented programing and object-oriented programing (OOP)
- **Python is formable:** python script can create console programs, graphical user interface (GUI) applications and common gateway interface (CGI) scripts to process web data.
- **Python is expandable:** it allows the addition of low-level modules to the interpreter for customization.

A python program is basically text file created with a python editor or such as windows Notepad, which has been saved with a “.py” file extension. Python program can be executed by stating the script file name after the python command at a terminal prompt. In python, the **print()** function is used to specify the message within it parentheses. This must be a string of characters together with this quote marks. These may be “ ” double quote marks or ‘ ’ singles quote marks but not a mixture of both.

### 8.2.1. CGI Programing in Python

CGI refers to as CGI is not a kind of language but just an arrangement or set of rules that helps to set up a dynamic interaction between a web application and the client application or browser. The CGI programs create feasible communication between client and web servers. Whenever the client browser sends a request to the web server the CGI programs send the output back to the web server based on the input provided by the client-server.

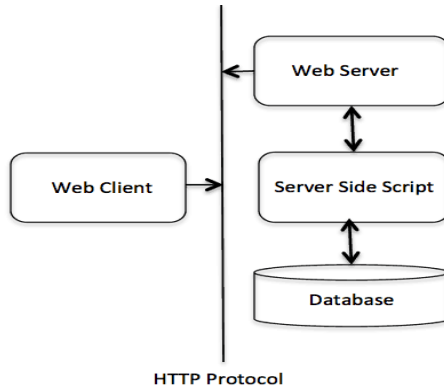
- CGI is the standard for client and web servers programs to interface with HTTP servers.
- CGI programing is written for generating web pages that respond to user input or web pages that interact with web application and the client application.

### 8.2.2. Working of CGI

In the CGI, once a request is prepared by the client-server to the web server, the CGI uses external script files to handle such requests. These files could be written in any language. The most important purpose of these script files

is to retrieve the data from the database promptly and more efficiently. These CGI scripts translate the retrieved data into an HTML format that sends the data to these web servers in HTML formatted page.

### 8.2.3. CGI Architecture Diagram



**Figure 8.1.** *CGI architecture.*

Some advantages of CGI:

- CGI are portable and can run or execute on almost any web server and operating system.
- CGI are Programming language-independent.
- CGI are relatively scalable programs in environment which means they can perform both simple and complex tasks.
- CGIs improve dynamic communication in web application and the client application.
- The CGI also help in making businesses more profitable by decreasing development and maintenance costs.

Some disadvantages of CGI:

- The CGI script has to evaluate by interpreter every time the program is initiated this creates a lot of traffic as there are too many requests from the side of the client-server.
- CGI programming and designing of is very difficult and ambiguous.
- CGI programs may cooperation server security as most of them are free and easily available making them quite vulnerable.

### 8.3. PYTHON WITH HTML

Whenever a user asks to view an online web page in their browsers it requests the page from the web server and receives the page in response via the HTTP protocol. When a requested web page address is an HTML document with an .html file extension the web server response will return that file to the browser so its contents can be displayed. Where python is installed on the computer hosting the web server. The web server can be configured to recognize Python scripts with a .py file extension and call upon the Python interpreter to process script code before sending an HTML response to the web server, for return to the browser client.

A Python scripts requested by a web browser can generate a complete HTML documents response by describing the content type on the first lines as **Content-type:text/html\r\n\r\n** so the web browser will parse the markup content for display on the screen. Ensure that the web server is running and configured to execute Python scripts. Python scripts save with a .py file extension and finally save the file in the web server's HTML document directory like **/htdocs**. Next run your program, in your browser with `http://localhost/file.py` then it executed.

#### 8.3.1. How to run Python Script on XAMPP

The Python script or code will execute or run as on web server or a command line interface. Python codes for the web need to install web server and need database server. There are various web servers for executing python program or code like XAMPP or WAMP and database server like MySQL.

Basic steps for execute Python Script on XAMPP Server:

Step 1. Download XAMPP from specific url and install XAMPP in your machine.

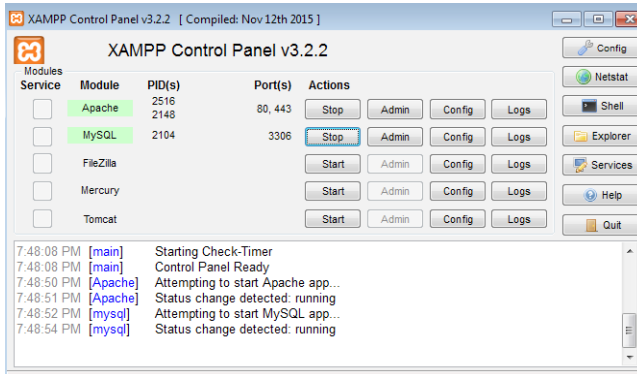
Step 2. After successfully installation of XAMPP server, use the XAMPP control panel to start / stop all servers.

Step 3. Write Python program in a notepad and save it with .py file name.

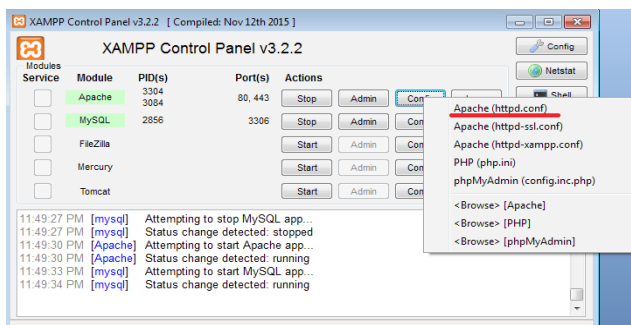
Step 4. Copy .py file inside htdocs folder (C:/Program Files/XAMPP/htdocs).

Step 5. Next start Apache servers and Mysql from the XAMPP control panel and get the dashboard for localhost: search `http://localhost` in any browser.





Step 6. Next configured to execute python scripts with click Apache servers Config in Apache (httpd.conf) and find addhandler, inside this file edit with .py and save this file as shown in figure. This is important to run python script on XAMPP server.



Edit inside file httpd.conf:

```

#
#AddType application/x-gzip .tgz
#
# AddEncoding allows you to have certain browsers uncompress
# information on the fly. Note: Not all browsers support this
#
#AddEncoding
#AddEncoding
#
# If the AddEncoding directive is commented-out, you can still use
# the Accept-Encoding header as a means for the browser to
# request higher compression (usually applied to text files).
#
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
#
#
# AddHandler allows you to map certain file extensions to
# actions unrelated to filetypes. These can be either built-in
# or added with the Action directive (see below)
#
# To use CGI scripts outside of ScriptAlias'ed directories
# (You will also need to add "ExecCGI" to the "Options"
# directive for those directories)
AddHandler cgi-script .cgi .pl .asp .py

```

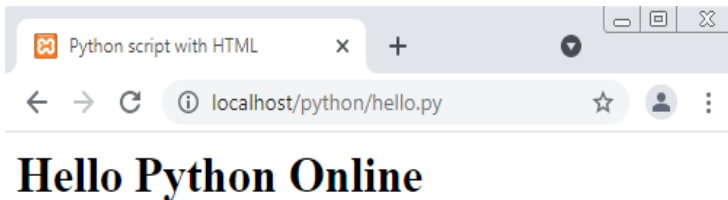
Step 7. Now run your program, in your browser with `http://localhost/file.py` then it executed.

Step 8. Output will display in your browser.

**Example: hell.py8.1**

```
#!C:/Python27/ArcGIS10.5/python.exe
print()
import cgi
print("<html>")
print("<head>")
print("<title> Python script with HTML </title>")
print("</head>")
print("<body>")
print("<h1> Hello Python Online </h1>")
print("</body>")
print("</html>")
```

Output:



**Figure 8.2.** *Output hello.py.*

## 8.4. SUBMITTING FORMS WITH PYTHON

Values can be passed to a python script on the web server when the browser makes an HTTP request. Those values can be used in the script and echoed in a response returned to the browser.

Python's "cgi" module can be used to easily handle data passed from the web browser by an HTTP request. This provides a **fieldStorage()** constructor that creates an object storing the passed data as dictionary of key: value pairs.

Any individual value can then be retrieved by specifying its associated key name within the parentheses of that FieldStorage object's **getvalue()** method. The browser can submit data to the script using a "GET" method that simply appends key=value to the script's URL address.

Passing data from a web page to a web server using the GET method to append key: value pairs to a URL is simple but has some limitation, the request string length cannot exceed 1024 characters and values appear in the browser address field.

As a more reliable alternative the browser can submit data to the script using a "POST" method that sends the information to the web server as a separate message not appended to the URL. Python "cgi" module can be used to handle form data sent from the browser with the POST method.

Example: post.html 8.2

```
<html>
<head>
<title> Python Appended Values </title>
</head>
<body>
<form method ="POST" action ="/cgi-bin/post.py">
Make: <input type ="text" name ="make" value ="ford">
Model: <input type ="text" name ="model" value ="mustang">
<input type ="submit" value ="Submit">
</form>
</body>
</html>
```

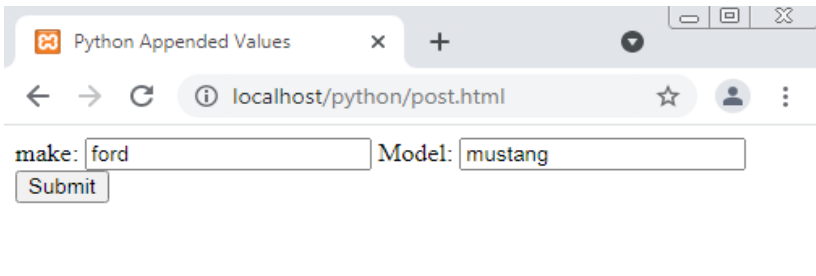
Next, start a new Python script by making CGI handling feature available and create a FieldStorage data object and assign two passed value to variables by specifying their associated key names. Then add statements to output an entire HTML web page including passed values in the output. Finally save html files in the web server's /htdocs directory and python script save in the web server's /cgi-bin directory.

**Python script Post.py**

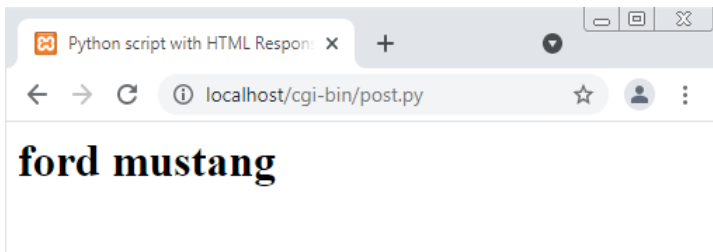
```
#!C:/Python27/ArcGIS10.5/python.exe
import cgi, cgiib
data =cgi. FieldStorage()
make =data.getvalue('make')
model =data.getvalue('model')
print("Content-type:text/html\r\n\r\n")
print("<html>")
print("<head>")
print("<title> Python script with HTML Response </title>")
print("</head>")
print("<body>")
print("<h1>,"make, model, "</h1>")
print("</body>")
print("</html>")
```

Output:

After click Submit button python script run and give output as figure 8.4.



**Figure 8.3.** *Output post.html.*



**Figure 8.4.** *Output post.py.*

## 8.5. PROVIDING TEXT AREA WITH PYTHON

Large amount of user-input text data can be passed from a web page to a web server using HTML `<textarea>` tags and the form POST method. This tag has no value attribute so a default value may not be provided. It is, therefore useful to have the python script test whether the text area has been left blank and provide a default value when the user has entered no text.

For example, create a new HTML script containing a form with a text area field and submit button.

Example: text.html 8.3

```
<html>
<head>
<title> Python text area example </title>
</head>
<body>
<form method =“POST” action =“/cgi-bin/text.py”>
<textarea name =“text web” rows= “5” cols=“20”>
</textarea>
<input type =“Submit” value =“Submit”>
</form>
</body>
</html>
```

Next, start a new python script by making CGI handling features available and create a Fieldstorage data object.

Python script text.py

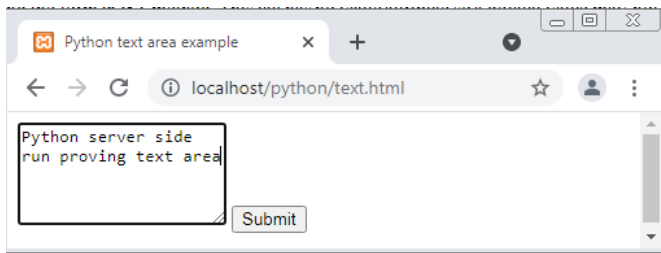
```
#!/C:/Python27/ArcGIS10.5/python.exe
import cgi, cgitb
data =cgi.FieldStorage()
text= data.getvalue('text web')
print(“Content-type:text/html\r\n\r\n”)
print(“<html>”)
print(“<head>”)
print(“<title> Python script with HTML Response </title>”)
```

```
print("</head>")
print("<body>")
print("<h1>")
print(text)
print("</h1>")
print("</body>")
print("</html>")
```

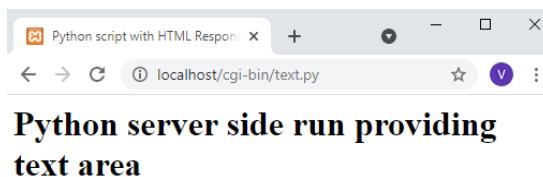
Finally save html files in the web server's /htdocs directory and python script save in the web server's /cgi-bin directory.

Output:

After click Submit button python script run and give output as Figure 8.6.



**Figure 8.5.** *Output text.html.*



**Figure 8.6.** *Output text.py.*

## 8.6. CHECKING BOXES WITH PYTHON

An HTML form can provide a visual checkbox “on/off” control that the user can toggle to include or reject its associated data for submission to the web server. The python script nominated to handle the form data can test

whether each check box has been checked simply by testing if a value has been received from the checkbox of that name.

For example, create a new HTML script containing a form with a two checkboxes and submit button.

#### **Example: checkbox.html 8.4**

```
<!DOCTYPE HTML>
<html>
<head>
<title> Python check box example </title>
</head>
<body>
<form action = "/cgi-bin/checkbox. py" method = "POST" target = "_
blank">
<input type = "checkbox" name = "C++" value = "on" /> C++
<input type = "checkbox" name = "python" value = "on" /> Python
<input type = "submit" value = "Select Language" />
</form>
</body>
</html>
```

Next, start a new python script by making CGI handling features available and create a FieldStorage data object.

#### **checkbox.py**

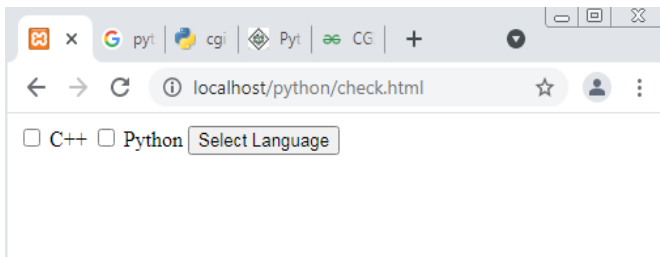
```
#!C:/Python27/ArcGIS10.5/python. exe
import cgi, cgitb
form = cgi.FieldStorage()
if form.getvalue('C++'):
C_flag = "ON"
else:
C_flag = "OFF"
if form.getvalue('python'):
py_flag = "ON"
else:
```

```
py_flag = "OFF"
print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>"
print "<title>Checkbox Program</title>"
print "</head>"
print "<body>"
print "<h2> CheckBox C++ is: %s</h2>" % C_flag
print "<h2> CheckBox Python is: %s</h2>" % py_flag
print "</body>"
print "</html>"
```

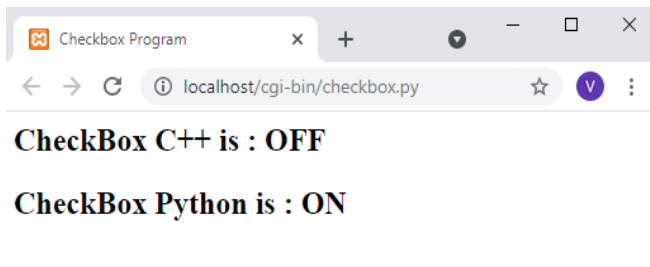
Finally save html files in the web server's /htdocs directory and python script save in the web server's /cgi-bin directory.

### Output:

After select check box python and click Submit button python script run and give output as Figure 8.8.



**Figure 8.7.** *Output checkbox.html.*



**Figure 8.8.** *Output checkbox.html.*



## 8.7. RADIO BUTTONS WITH PYTHON

An HTML form can provide a “radio button” group from which the user can select just one button to submit its associated data to the server. Unlike checkboxes, radio buttons that share a common name are mutually exclusive so when one button in the group is selected, all other buttons in that group are switched off. The python script nominated to handle the form data can test the value submitted for the radio button group name and supply an appropriate response.

For example, create a new HTML script containing a form with a two radio buttons and submit button.

Example: radiobutton.html 8.5

```
<html>
<head>
<title> Python radio buttons example </title>
</head>
<body>
<form action = “/cgi-bin/radiobutton.py” method = “post” target = “_
blank”>
<input type = “radio” name = “subject” value = “C++” /> C++
<input type = “radio” name = “subject” value = “python” /> Python
<input type = “submit” value = “Select Language” />
</form>
</body>
</html>
```

Next, start a new python script by making CGI handling features available and create a Fieldstorage data object.

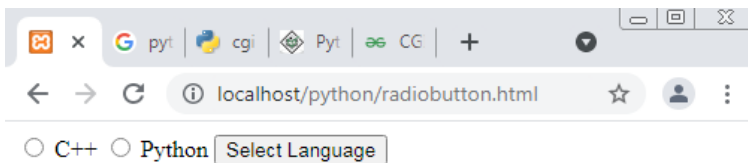
**radiobutton.py**

```
#!C:/Python27/ArcGIS10.5/python.exe
import cgi, cgitb
form = cgi.FieldStorage()
if form.getvalue(‘subject’):
    subject = form.getvalue(‘subject’)
else:
```

```
subject = "Not set"
print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>"
print "<title>Radio Program</title>"
print "</head>"
print "<body>"
print "<h2> Selected Subject is %s</h2>" % subject
print "</body>"
print "</html>"
```

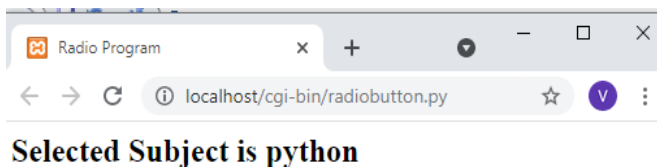
Finally save html files in the web server's /htdocs directory and python script save in the web server's /cgi-bin directory.

Output:



**Figure 8.9.** *Output radiobutton.html.*

After tick radio button python and click Submit button python script run and give output as Figure 8.10.



**Figure 8.10.** *Output radiobutton.py.*

## 8.8. DROP DOWN (SELECT) OPTION WITH PYTHON

An HTML form can provide a dropdown list of possible options from which the user can select a single option to include its associated data for submission to the web server. The submitted value can then be retrieved by specifying its associated list key name within the parentheses of that fieldStorage object's `getvalue()` method. This is used when we have several options available but only one or two will be selected.

For example, create a new HTML script containing a form with a dropdown options list and submit button.

Example: dropdown.html 8.6

```
<!DOCTYPE HTML>

<html>
<head>
<title> Python drop down example </title>
</head>
<body>
<form action = "/cgi-bin/dropdown.py" method = "post" target = "_
blank">
<select name = "dropdown">
<option value = "Java" selected>Java</option>
<option value = "Python">Python</option>
<option value = "C++">C++</option>
</select>
<input type = "submit" value = "Submit"/>
</form> </body>
</html>
```

Next, start a new python script by making CGI handling features available and create a FieldStorage data object.

dropdown.py

```
#!/C:/Python27/ArcGIS10.5/python.exe
```

```
import cgi, cgitb
```

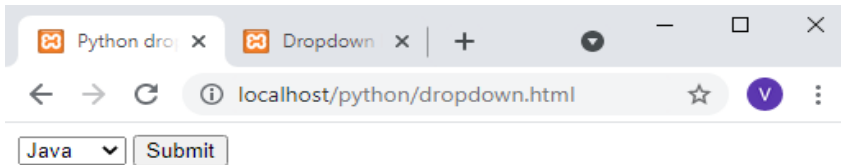
```
form = cgi.FieldStorage()
```

```
if form.getvalue('dropdown'):
```

```
subject = form.getvalue('dropdown')
else:
subject = "Not entered"
print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>"
print "<title>Dropdown Box Program</title>"
print "</head>"
print "<body>"
print "<h2> Selected programing language is %s</h2>" % subject
print "</body>"
print "</html>"
```

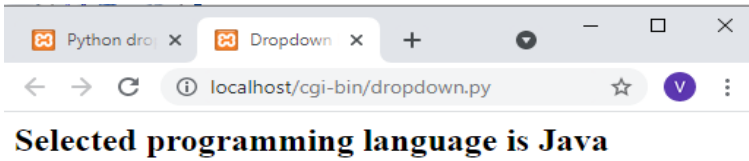
Finally save html files in the web server's /htdocs directory and python script save in the web server's /cgi-bin directory.

Output:



**Figure 8.11.** *Output dropdown.html.*

**After select java and click Submit button python script run and give output as Figure 8.12.**



**Figure 8.12.** *Output dropdown.py.*

## 8.9. UPLOAD FILES WITH PYTHON

An HTML form can provide a file selection facility, which calls upon the operating system’s “Chrome file” dialog, to allow the user to browse their local file system and select a file. To enable this facility the HTML `<form>` tag must include an `enctype` attribute specifying the encoding type as “`multipart/form-data`.”

The full path address of the file selected for upload is a value stored in the `FieldStorage` object list that can be accessed using its associated key name. A copy of an uploaded file can be written on the web server by reading from the `FieldStorage` object’s `file` property.

For example, create a new HTML script containing a form with a file selection facility and a submit button.

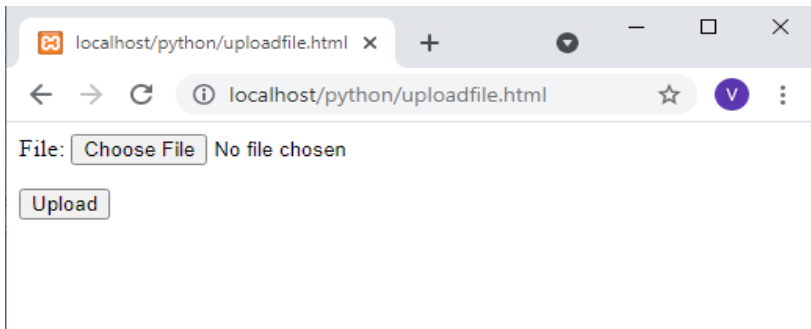
### **Example: uploadfile.html 8.7**

```
<!DOCTYPE HTML>
<html>
<body>
  <form enctype = "multipart/form-data" action = "/cgi-bin/upload. py"
method = "post">
    <p>File: <input type = "file" name = "filename" /></p>
    <p><input type = "submit" value = "Upload" /></p>
  </form>
</body>
</html>
upload.py
```

```
#!C:/Python27/ArcGIS10.5/python. exe
import cgi, os
import cgitb; cgitb.enable()
data = cgi.FieldStorage()
# Get filename here.
upload = data['filename']
fn = os.path.basename(upload.filename)
open('/tmp/' + fn, 'wb').write(fileitem.file.read())
print ("content-type:text/html\r\n\r\n")
print("<html>")
print("<head>")
Print("<title> Python upload file</title>")
print("</head>")
print("<body>")
print("<h1> File uploaded," filename, "</h1>")
print("</body>")
print("</html>")
```

Finally save html files in the web server's /htdocs directory and python script save in the web server's /cgi-bin directory.

### Output:



**Figure 8.13.** *Output uploadfile.html.*

After click choose file select a vky.jpg file and click Upload button python script run and give output as figure 8.14.

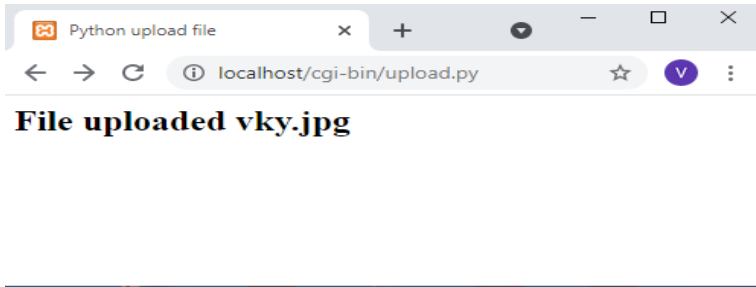


Figure 8.14: Output *upload.py*.

### Summarize:

- Python can be installed on a web server host to process script code before sending a response to a web browser client.
- Through describing the content type as **Content-type:text/html\r\n\r\n**, server-side python script can generate an HTML document.
- The **cgi** module provide a **FieldStorage()** constructor to create an object for storing submitting data as key:value pairs.
- Through specifying its key name to the objects's **getvalue()** method, any value stored in a FieldStorage object can be retrieved.
- Using the GET method the browser can send data to a script that appends key=value pairs to its URL address after? mark.
- **GET** method request string length cannot be exceeding 1024 characters and it will be visible in the browser address field.
- The browser can send data to a script using the **POST** method that submits key=value pairs as separate message.
- Data submitted from an HTML form can be stored in a **FieldStorage** object as key=value pairs for each form field.
- In the Checkbox fields of an HTML form that are unchecked do not obtain submitted to the web server.
- In the selected radio button in a group provides the value to be connected with the group name when the form gets submitted.
- In the selected item in a dropdown list provides the value to be connected with the list name when the form gets submitted.
- File uploads only if it **enctype** attribute specifies it encoding type as "**multipart/form-data**" that can allow HTML form.





# Index

---

## A

Accept header fields 10  
Administration 217  
advanced research projects agency  
  network (ARPANET) 5  
Analytic software 27  
Application 105  
Application programing interface  
  (API) 138  
Architecture 2, 3, 8, 16  
Argument 203  
Array 173, 174, 175, 176, 177, 180,  
  184, 190  
Attribute 44, 45, 47, 49, 52, 54, 55,  
  56, 57, 59, 64, 67, 71, 95  
Audio conference 5

## B

Background-repeat property 72, 74  
Border-color property 88  
Browser 2, 4, 7, 10, 13, 14, 18, 33,  
  39, 40, 54  
Browser compatibility 18  
Business requirement 23

## C

Capability 113  
Cascading style sheets 62, 103  
Checkbox 231, 232  
Chrome 220, 237  
Client-side JavaScript 108  
Client-side markup language 141  
Collect information 58  
Common gateway interface (CGI)  
  14, 222  
Computer application 191  
Confirm box 127  
Confirm Popup box 129  
Constructor 133, 135  
Content delivery network (CDN) 14  
Control dynamic 166  
Cookie value 189  
Current array element 174  
Customer 217

## D

Data 150, 151, 156, 158  
Databases 200, 217, 218  
Design procedure 22

Design process 16  
 Desktop publishing (DTP) 21  
 Display 203, 204, 206, 208, 210,  
 211, 213, 215, 217  
 Document 32, 38, 39, 44  
 Document object model (DOM) 138  
 Domain name server (DNS) 4  
 Dropdown 235, 236, 237, 239  
 Dynamic Hypertext Markup lan-  
 guage 141, 142  
 Dynamic web pages 108, 142

## E

Efficiency 166  
 Efficient retrieval 200  
 Element 65, 66, 67, 69, 70, 71, 72,  
 73, 78, 81, 85, 86, 88, 91, 93,  
 94, 95, 96, 97, 98, 101, 102,  
 103, 105  
 Element-declarations 155  
 Element relative 101  
 Element Syntax 152  
 Entity 153, 155  
 Essential data-type 132  
 Extensible markup language 150  
 Extensible stylesheet language  
 (XSL) 159  
 External source 155, 156  
 External style sheet 69, 70

## F

Familiarity 166  
 Firefox 220  
 Fixed positioning 102  
 Flexibility 166  
 Flexible box 104  
 Frequently 122, 127, 129, 130  
 Function 111, 112, 113, 114, 115,

116, 128, 129, 131, 132, 133,  
 134, 135, 140, 144, 146

## G

Gain publicity 24  
 Global change 63  
 Global scope 111  
 Good software development 23  
 Gradient function 105  
 Graphical user interface (GUI) 222  
 Greenwich Mean Time (GMT) 9

## H

Hypertext Preprocessor 166  
 Hypertext transfer protocol (HTTP)  
 2, 7

## I

Information 2, 5, 6, 7, 8, 11, 12, 14,  
 15, 16, 17, 19, 20, 24, 25, 26,  
 27, 28, 29  
 Informix 166  
 Inline style 71  
 Instance 195, 197  
 Internet assigned number authority  
 (IANA) 10  
 Internet Explorer 220  
 Internet media 10  
 Internet protocol 5  
 Inter-related data 200  
 Inventory 24

## J

JavaScript 62, 104, 105, 107, 108,  
 109, 110, 111, 112, 115, 116,  
 117, 120, 122, 123, 124, 125,  
 127, 128, 129, 130, 132, 133,  
 136, 139, 141, 142, 143, 144,  
 145, 146, 147

JavaScript frameworks 14  
 JavaScript function 115, 144  
 JavaScript technology 141, 143

## L

Language-neutral interface 138  
 Linux 167  
 Loop iteration 174

## M

Management 217  
 Management system 27  
 Manufacturer 29  
 Margins 93  
 Marketing goal 25  
 Markup language 32  
 Mathematical analysis 220  
 Media queries 104, 105  
 Multi-dimensional array 177  
 MySQL management 217

## N

Number specifying space 81

## O

Object-oriented facility 108  
 Object-oriented programing (OOP) 194, 222  
 Online publishing 27  
 Operating system's 237  
 Optional return statement 115  
 Organization 6, 25, 28, 150

## P

Paragraph 63, 68, 69, 70, 71, 72, 78, 79, 80, 82, 96, 97, 98  
 Perform operation 194  
 Perform special 116

Possible values 81, 82, 83, 85, 86  
 Procedural programing 194  
 Programing interface 138, 139  
 Prompt popup box 130  
 Python interpreter 224  
 Python language 220, 221  
 Python programing language 220, 221  
 Python script 224, 226, 227, 228, 229  
 Python web script 220

## Q

Query-by-example (QBE) 218

## R

Radio buttons 233  
 Resource Description Framework (RDF) 3  
 Retrieve relevant information 161  
 Return value 170  
 Reusable program 112  
 Root element 153, 155, 156, 161

## S

Security 166  
 Selection facility 237  
 Server-side scripts program 108  
 Server-side validation 139  
 Session tracking 166  
 Several platform 200  
 Simplicity 166  
 Single property 76  
 Social media 14, 25  
 Software Engineer 62  
 Specific element 161  
 Standard generalized markup language (SGML) 150

String 180, 181, 184, 186

Submit button 182

Subtract space 78

## T

Transfer volumes 6

Transform 105

Transformation 159

Transform-origin 105

## U

Uniform resource identifier (URI) 2

Uniform resource locator (URL) 2

Unique function 112

Unique style 70, 71

Universal selector 66

User-centered design (UCD) 22

User-input text data 229

## V

Value 109, 110, 113, 114, 115, 116,  
120, 122, 130, 132, 133, 137,  
140, 146

Variable initialization 110

Visual checkbox 230

## W

Web browser 64, 108, 147, 220,  
224, 227, 239

Web designer 62

Web developer 220

Web development 220

Web document 64

Web language 108

Web Ontology Language (OWL) 3

Web programing language 108

Web publishing 27

Web server 222, 223, 224, 226, 227,  
229, 230, 232, 234, 235, 236,  
237, 238, 239

Website development software 27

World Wide Web Consortium (W3C)  
150

World Wide Web (WWW) 2



# Fundamentals of Web Technology

The book highlights the fundamentals of web page development and prepares students to make real-world, industrial, or business strength web-based applications, and use a wide variety of Web development tools effectively and efficiently. Web technology is necessary for today because the internet has become the number one source of information, and many of the conventional software applications have become web applications. Web applications have become more strong and can fully replace desktop applications in most situations. Web technologies are the several tools and methods that are applied in the activity of communication between different types of devices over the internet. The basic principle, which will cover web browsers and some web app creation programming languages and frameworks which are used in the creation of websites. Data or information collected by the websites to store and access store data at the back-end is used databases connectivity programing. Hypertext markup language, referred to as HTML, is where the WWW started. It is a basic instruction that covers the basics concept of the web. CSS is one of the most important website design technologies that provide look and fill facilities for your web pages, and you can merge cascading style sheets (CSS) into your HTML code. JavaScript help to design interactive web pages, the involvement of graphical elements, database integration, and dynamics of your website. PHP stands for hypertext preprocessor. It is a scripting language preferred to use by both beginners and professionals in the web development areas.

This book reflects the author's experience in teaching courses on internet and web technology for more than 12 years. This book provides students and web developers with an understandable introduction to the web programming and scripting languages used to create Web sites and web applications. The main aim is to teach the programming concepts of different Web technologies and the fundamentals needed to program on the internet. We cover in this textbook fundamentals of internet, world wide web (WWW), HTML, CSS, JavaScript, XML, PHP, and connectivity with database using MYSQL and also cover the introduction of the web with python and CGI scripting to manage web from data. The contents of the book have been thoroughly organized and spread over eight chapters. Each chapter begins with the basics introduction and describes with syntax, code, result output, diagrams, and examples. We trust this textbook would meet the rich practical hands-on experience in developing web applications, combined with teaching the subject for graduate/post-graduate students and who want to study the internet and web technology. We would very much appreciate receiving any suggestions for improvement in the book from teachers, students, and other readers. We wish to express our deep thanks to all those who helped in making this book a reality. We express our gratitude to our publisher and the team of publications that have taken great pains in publishing the book with speed and accuracy.



**Dr. Anil Kumar Yadav** received his PhD in Computer Science and Engineering and his M.Tech in Information Technology, B.Tech in Computer Science and Engineering. Presently, he is working as an Associate Professor in Computer Science & Engineering Department at IES College of Technology Bhopal M.P. His major areas of interest include Application-Oriented Reinforcement Learning, Game Technology and Machine Learning. He received many academic related awards. He has published 03 Patent, one text book on "Data Structures with C program," ARCLER PRESS and more than 34 research publication at National and International Level.



**Mr. Vinod Kumar Yadav** is pursuing PHD (CSE) from RNTU, Bhopal and completed his M.Tech in Computer Science and Engineering from the University of Rajiv Gandhi Proudhyogiki Vishwavidyalaya Bhopal M.P. and B.Tech in Information Technology from Guru Ghasidas Vishwavidyalaya Central University Bilaspur C.G. (India). He is also GATE Qualified in Information Technology. At present he is working as Head of Department ,Computer Science & Engineering BMCT, Bhopal (M. P.). His areas of interests are in Data Structures, Data mining, Artificial Intelligence, Machine Learning, Deep Learning and Computer Vision. He has supervised M.Tech and B.Tech students. He received many academic related awards. He has published a Book "Data Structures with C Programming" ARCLER PRESS, 2010 Winston Park Drive Oakville Ontario L6H 5R7 Canada in 2019. He has published 12 research papers at National, International Conferences and International Journals. He also attended several National & International workshops.