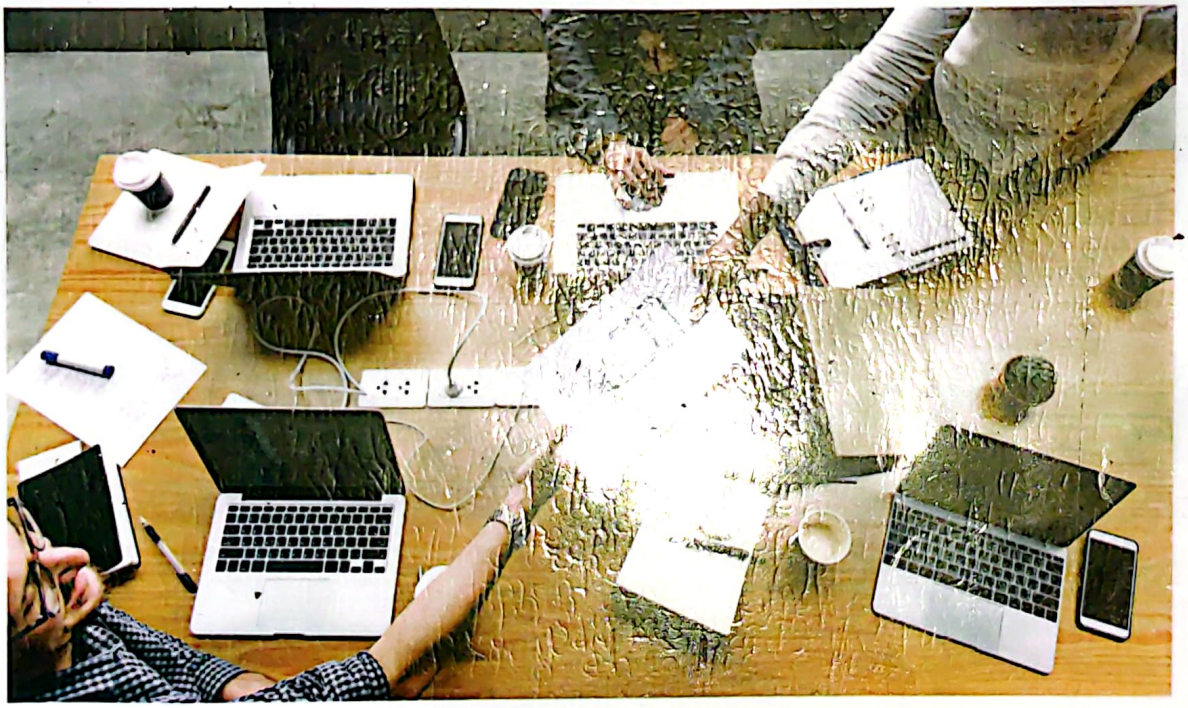
D.E

INTRODUCTION TO WEB PROGRAMMING



A LABORATORY MANUAL

Victor E. Ekong
Uyinomen O. Ekong
Patience U. Usip
Ifiok J. Udo
Kingsley U. Joseph

Department of Computer Science
Faculty of Computing
University of Uyo, Nigeria.

Aim and Objectives

The aim of this manual is to enhance the practical skills of the students in web site development. The objectives are to:

- Introduce the student to the basic concept, features and services of the Internet and the world-wide-web (WWW).
- Describe the architecture of web applications and the tools for developing standard websites
- Describe the client-server model of web applications
- Demonstrate the use of HTML tags and attributes for web page designs
- Demonstrate the use of cascading style sheets (CSS) for imposing style on the content of HTML tags
- Demonstrate the use of JavaScript for adding interactivity to HTML web pages
- Demonstrate how to create extensible markup language (XML) documents that will describe and validate web documents
- Describe the web development life cycle and the roles involved in web development

Table of Contents

Module 1
Web Programming Basics

Unit 1: Definitions of Web Programming, the Internet and World Wide Web (WWW)

This module begins by introducing some theoretical concepts of the Internet and the Web. It explains the importance and the need for web programming. It elaborates on the Internet and the World Wide Web programming models.

What is Web Programming?

Web programming is the process of writing codes that will eventually create a website. Texts and multimedia contents can be manipulated by a web designer to produce an elegant page that is seen on a web browser. Web programming is therefore the creation of web pages that run over the Internet.

Importance of Web Programming

Web programming is important due to the following facts:

***Global Reach*:**

The Internet and the web allow businesses to reach a worldwide audience with their products and services.

***Cost-Effectiveness*:**

Web development enables businesses to reduce costs by automating processes, improving efficiency, and reducing the need for physical infrastructure.

***Improved Communication*:**

Websites facilitate communication and collaboration across organizations and geographies, enabling better knowledge sharing and decision-making.

***Enhanced Customer Experience*:**

Web applications allows businesses to provide a more personalized, interactive, and engaging customer experience.

***Access to Data*:**

Web applications provides vast amounts of data that can be used for analysis, insights, and informed decision-making.

***Mobile Access*:**

Web applications enables businesses to reach customers through mobile devices, which have become increasingly important our lives.

***Innovation*:**

Web applications drive innovation by providing a platform for new products, services, and business models.

***Social Impact*:**

Web technology can improve social and economic outcomes by providing access to education, healthcare, and other essential services.

In conclusion, web programming is a critical enabler of the digital economy, driving growth, innovation, and social progress.

Why must I learn Web programming?

***In-Demand Skills*:**

Web development skills are in high demand, with many companies seeking professionals who can design, develop, and maintain web-based applications.

***Career Opportunities*:**

Learning web programming can open up many career opportunities, including web development, web design, e-commerce, and digital marketing.

***Flexibility*:**

Web programming can be used to build various applications and services, providing flexibility in the type of work that can be undertaken.

***Entrepreneurship*:**

Web programming provides a platform for entrepreneurship, enabling individuals to create and launch their web-based businesses.

***Creativity*:**

Web programming allows for creative expression through web design and development, providing opportunities to create visually appealing and engaging user interfaces.

***Continuous Learning*:**

Web programming constantly evolves, providing continuous learning and skill development opportunities.

***Remote Work*:**

Web programming skills are highly transferable and can be applied remotely, providing remote work and freelancing opportunities.

Competitive Advantage:

Knowledge of web programming can provide a competitive advantage in the job market and business, enabling individuals and companies to stay ahead of the curve.

What is the Internet?

The Internet is a global communication system consisting of hardware and software infrastructures that provide connectivity between remote computers.

What is the Web?

The Web is a collection of interconnected documents and other resources, linked by hyperlinks and universal resource locators (URLs). The World Wide Web is a hypermedia system. Hypermedia is the addition of multimedia to hypertext. A combination of text, graphics, video or sound can be easily interlinked in hypermedia document to offer a rich, often interactive, environment on the Web.

The World Wide Web

The World Wide Web (WWW) is a hypermedia system. Invented by Tim Berners-Lee in 1994, it is a global collection of interconnected documents on the Internet. The WWW is a part of the Internet that uses hypertext transport protocol (HTTP) to display Hypertext and images in a graphical environment. It is a hypertext based information retrieval tool. One can easily surf the web by jumping from one document to another using the links in those documents. These documents can be in many formats, such as text, graphics, animation, sound and video. They may also be a combination of all these. All the information on Internet are presented to the user as a document or as a web page. The web pages are linked to each other or even to a section within a web page and these links are known as hyperlinks. WWW accesses resources available on disparate or similar network via HTTP. The tool used to view the web pages on Internet are called Internet browsers. This is a software program specifically developed to extract information on user requests from the Internet and present them as web pages to the viewer.

Internet Address

Just like every house, office, location has an address, every web page on the Internet has a unique address. This address is used to get the web page for users from the Internet. Just as the address of a house is known as its postal address, the address on the Internet is known as a

uniform resource locator (URL). A typical secure Internet address or URL would look like; https://www.uniuyo.edu.ng/fac-of-computing/se/victoreekong.htm. The URL locates a particular web page, among all the computers connected to the Internet. The URL contains the components that specify the protocol, server and pathname of an item. For the URL; https://www.uniuyo.edu.ng/fc/se/victoreekong.htm, the protocol is followed by a colon (HTTP:), the server is preceded by two slashes (//www.uniuyo.edu.ng), and each segment of the pathname is preceded by a single slash (/fc/se/victoreekong.htm). A protocol is a set of rules that tells the computer to know how to interpret the information at that address.

The first component, the protocol, defines the manner for interpreting computer information. Many web pages use HTTP. Other protocols used on the Internet include File Transfer Protocol (FTP) for file transfer, Simple Mail Transfer Protocol (SMTP) and Post Office Protocol 3 (POP3) for sending electronic mails.

The second component, the server (www.uniuyo.edu), identifies the computer system that stores the information you seek and is always preceded by two slashes. A server is a computer that has information stored on it and sends it to the client, when a request is made. Each server on the Internet has a unique address name whose text refers to the organization maintaining the server, in this example, University of Uyo.

The last component (/fc/se/), defines the path: faculty of computing/software engineering within the server (uniuyo.edu.ng) where the requested item (victoreekong.htm) will be found. Most web pages will have .htm or .html as their extension name.

How to Connect to the Internet

There are three (3) ways of connecting to the Internet today:

1. Connecting using Wireless Broadband (Wi-Fi):
   Connect to Wi-Fi by going to your network settings, turning on Wi-Fi, and selecting your network name.
2. Connecting using an Ethernet cable:
   Connect to Ethernet by using an Ethernet cable to connect your computer to your router or modem.
3. Connecting a computer using a Dial-up:
   Connect to dial-up by plugging in your modem to the phone jack, then connecting the modem to your computer.

Steps for Connection to the Internet

1. Ensure that the source of the Internet is on
2. Understand that mobile devices can only connect to wireless broadband (Wi-Fi). Devices like smartphones, tablets, iPods, hand-held devices can only connect to Wi-Fi services due to their portable nature. They cannot access the Internet through dial-up or Ethernet. These are limited to non-portable devices and computers as shown in Figure 1.

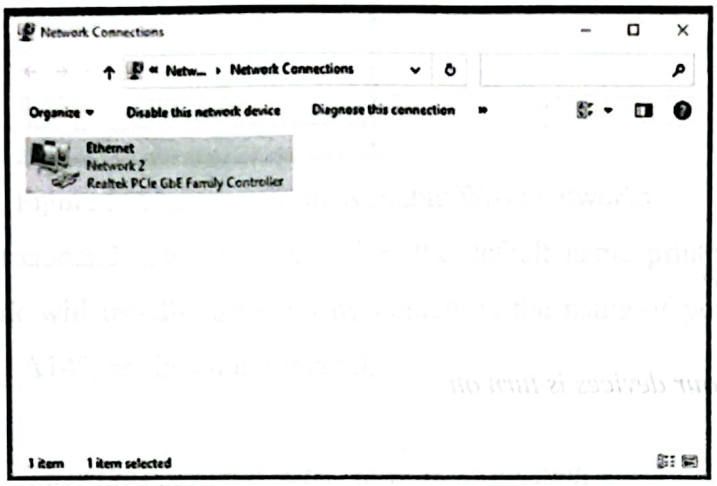


Figure 1: Ethernet Internet connection

3. Know the path to follow to get to the network setting. Regardless of the operating system or devices you are using, to access your network setting can be as follows:
   - Windows 7: Start =>Control Panel=>Network and Internet
   - Windows 8: Start=>Search "View Network Connections"=> View Network Connections
   - Windows 10/11: Search "View Network Connections"=> View Network Connections
   - Mac OS X: System Preference => Network
   - Linux, Ubuntu or Fedora: Network Manager
   - iOS (iPhone, iPad, etc.): Setting=> Wi-Fi
   - Android: Settings=>Wi-Fi (or Windows & Network)
   - Windows Phone: Settings=> Wi-Fi

Steps for Connections using Wireless Broadband (Wi-Fi)

1. *Find your devices network setting.*

   This is a wireless Internet connection usually done through a device's setting. The Wi-Fi option can be found in the settings menu of mobile devices and on taskbar on computers. This is where we can turn on the Wi-Fi and connect to a network. Example network settings locations include:

- Windows – Click the on Wi-Fi icon in the taskbar to switch on Wi-Fi.
- MacOS – Click the Wi-Fi icon on the menu bar to switch on Wi-Fi.
- Android – Go to Settings=>Network & Internet => Internet to turn on Wi-Fi
- iOS – Go to Settings => Wi-Fi to turn on Wi-Fi.

Figure 2: Network settings

2. *Make sure the Wi-Fi on your devices is turn on.*

Figure 3: Ensuring the Wi-Fi is turned on

3. *Go to the list of nearby Wi-Fi networks*

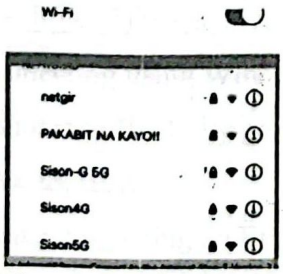


Figure 4: Selecting from a list of nearby Wi-Fi networks

The Wi-Fi menu on your device will have a list of nearby networks that you can connect to as shown in Figure 4.

4. *Find the name of your Wi-Fi network*

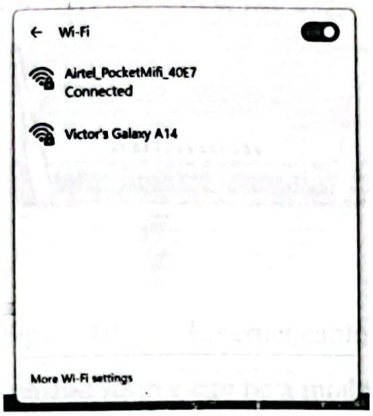


Figure 5: Choosing from available Wi-Fi networks

The broadband network's router has the default name printed on it. The name of a hotspot network will usually show up by default as the name of your cellular device (e.g. "Victor's Galaxy A14") as shown in Figure 5.

5. *Enter the password to the network or hotspot.*

Some networks are public, but most are not. If the network you are trying to connect to have a password, you will be prompted for that password before you can connect to the network. The default password will usually be listed on the router, but if you do not know the password, ask the person in charge of the network. Some protected public networks may have varying passwords per person. For example, a University may allow students to log on to the network with their student ID number, rather than a single set password.

Figure 6: Provide the network's password

6. *Wait for the network to connect*

It often takes a few seconds for a device to connect to a wireless source, but if the computer cannot establish the connection to the router, it will time out the connection attempt. In this case, move closer to the source, or disconnect and then reconnect your computer to the Wi-Fi. Figure 7

shows a connected network. However, connection attempt will fail if the password is not entered correctly.

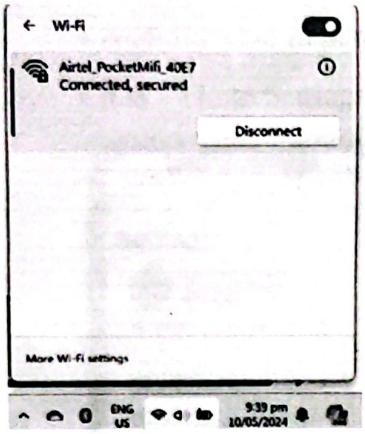


Figure 7: The connected network

7. *Test your Internet connection*

Once you have connected to the Internet, open up a page in your web browser and wait for it to load. Since some pages can crash, you may want to load up a reliable website, such as https://www.google.com/, as shown in Figure 8, to ensure that the website is not going to be down.

Figure 8: Opened web page

Steps for Connecting to the Internet using Ethernet Connection

1. *Get an Ethernet cable and any needed adapter*

Figure 9: An Ethernet cable

Ethernet cables (E.g. Cat5, Cat 5e or Cat 6) can be used depending on the speed of the network as shown in Figure 9.

2. *Connect one end of the Ethernet cable to a broadband Source*

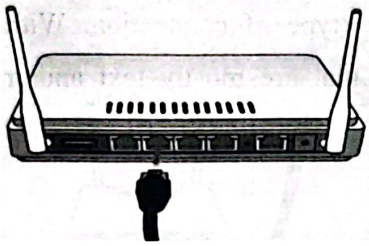


Figure 10: Ethernet cable connected to a router or broadband source

The broadband source can be a modem or a router as shown in Figure 10.

3. *Connect the other end of the cable to the computer*

The other end is connected to your laptop or computer devices through the Ethernet jack on the computer (port) as shown in Figure 11.

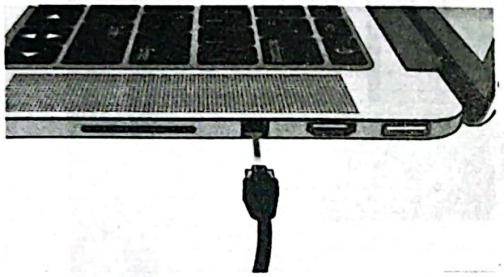


Figure 11: Ethernet cable connected to device network port

4. *Access the computer's setting and connection*

Ensure that the computer is set to recognize the Ethernet, rather than wireless. Most likely, you may have to turn off your wireless connection to ensure that the computer recognizes the Ethernet connection instead. Then test by opening up a web browser to see if it loads. You can load google to ensure that the connection is running as shown in Figure 12.

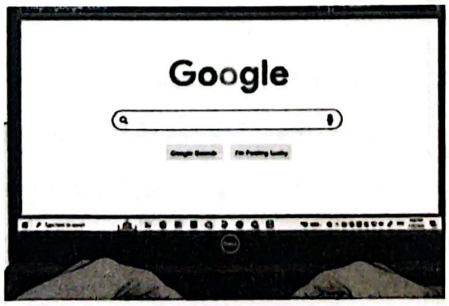


Figure 12: Loading a web browser to test the connection

Steps for Connecting to the Internet using Dial-up Connection

1. *Understand that Dial-up Internet is no longer widely supported* and it will be very difficult to do certain activities on the Internet with this type of connection. With dial-up Internet, you may be only limited to browsing websites that are mostly text and/or images without many add-ons and features.

2. *Ensure that you can connect to dial-up.*

It requires the use of a phone line and can only connect one person per phone at a time. If the phone is being used to make a call, the connection will be broken until the other person hangs up the call. Current computers do not have the components to connect to dial up, so it may require an external universal serial bus (USB), a land line phone and a computer with a modem jack.

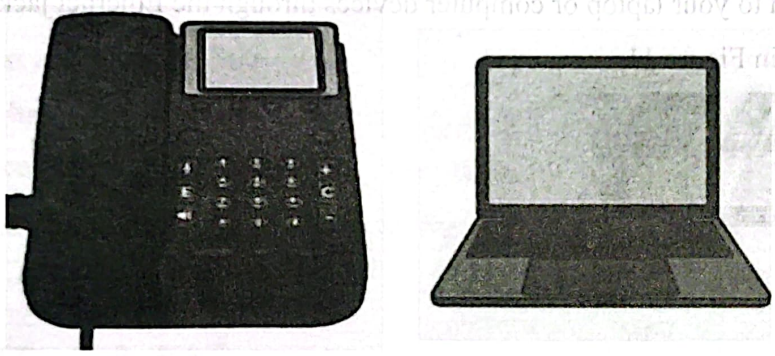


Figure 13: Required devices for dial-up connection

3. *Plug in the modem to the phone jack*

Ensure that the phone cable is plugged into both the phone jack on the wall and the plug on the modem as seen in Figure 14.

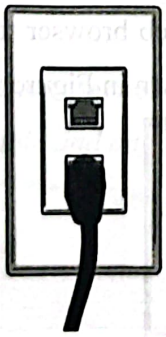


Figure 14: The phone jack plugged to the modem

4. *Connect the Modem to the Computer*

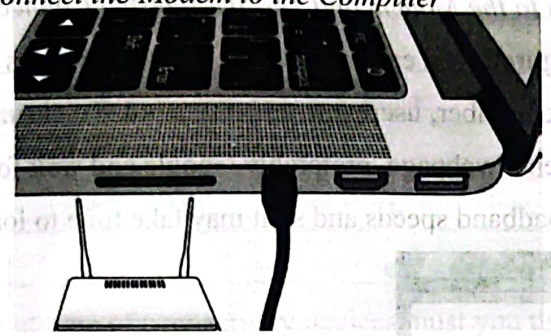


Figure 15: The modem is connected to the computer

Insert one end of the phone cable into the modem and the other end into the computer's modem jack as shown in Figure 15. The phone jack on the computer should be noted by a small phone next to it.

5. *Access the Computer's Network Settings*

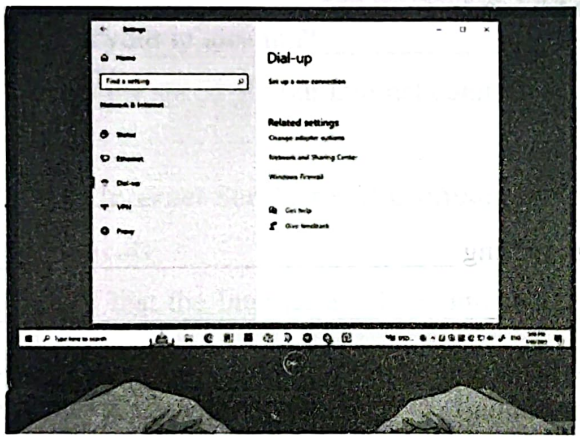


Figure 16: Accessing the computer's network setting

To manually configure the dial-up connection on the computer and the modem network settings as shown in Figure 16, the underlying operating system suffices. The settings path to follow for the configuration of the network are as per their operating systems;

- Windows 7 and 8: Network and Internet => Network and Sharing Center => Set up a new connection or network => Connect to the Internet => Dial-up
- Windows 10/11: Network => Dial-up Connection
- Mac OS X: Network => Internal/External Modem => Configuration
- Linux, Ubuntu or Fedora: Network Manager => Connections => Modem Connections => Properties

7. *Connect the Computer's connection to the Modem and Test the Internet Connection*

Where the dial-up setting are already configured, you can open up the network settings and connect to the modem by proving the phone number, username and password. To ensure that the Internet connection is working, you can open a webpage, preferably Google and wait for it to load. Dial-up Internet is slower than the broadband speeds and so it may take time to load.

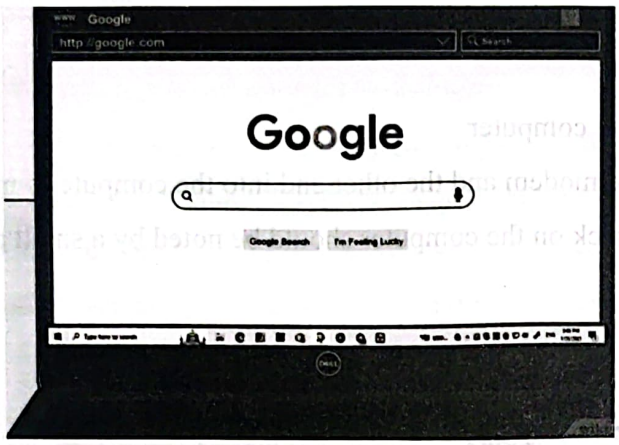


Figure 17: Loaded Google web page

Practical Exercise 1

1. In your own words, define web programming _____

_____

_____

2. List 5 benefits of studying web programming

a. _____

b. _____

c. _____

d. _____

e. _____

3. What is the difference between WWW and the Internet?

_____

_____

4. Write the full meanings. a. Wi-Fi _____ b. FTP _____

c. HTTP _____ d. POP _____

d. SMTP _____ e. URL _____

5. What is a website? _____

6. What is a web page? _____

_____

7. What is a web browser? Name five? _____

_____

a._____ b._____ c._____

d._____ e._____

8. What type of connectivity devices must you use to access the Internet through a public switched telephone network (PSTN)? _____

9. State the steps for connecting to the Internet using your android phones, laptop and a personal computer (PC)?

10. Connect to the Internet. Use your web browser and your favorite search engine. Type "For IP to Location, IP Tracing, Broadband Speedtest".

What is your ip-address? _____

What is your ip location? _____

What is the speed of your internet connection?_____-_____

UNIT 2: Internet Services and Communication Protocols

Internet Protocols

We understand that the Internet is a large interconnection of computers across the globe linked together through telecommunication networks. Communication on the Internet is guided by protocols. So a network protocol is a standard way of regulating data transfer between computers: Just like diplomats adhere to protocols-rules of behavior when in a foreign land, network communication is the same. They have to obey agreed rules if they are to communicate with each other. For two computers to communicate with each other, these protocols must be established as shown in Figure 18.

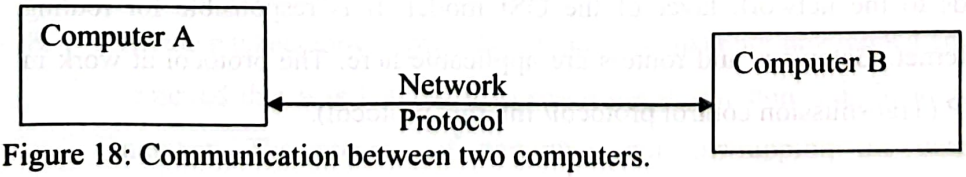


Figure 18: Communication between two computers.

Internet Model

This is an ARPANET model different from the open system of standard communication (OSI) model. The OSI model is a seven layer architecture for computer network communication

composed of: physical layer, Data link layer, Network layer, Transport layer, Session layer, Presentation layer and Application layer. The Internet model is composed of the Network access layer, Internetwork layer, Host-to-Host layer and Application layer as shown in Figure 19.

Application	7
Presentation	6
Session	5
Transport	4
Network	3
Data Link	2
Physical	1

Application	4
Host-to-Host	3
Internetwork	2
Network access	1

Figure 19: OSI Network model and the Internet Model

Application Layer

This layer combines the functionality of the topmost 3 layers of the OSI (Session, Presentation and Application layers). The protocol at this layer includes: Telnet, e-mail, Directory services (NFS), Network management services.

Host-To-Host Layer

This layer is equivalent to the transport layer of the OSI model. The protocol at this layer are transmission control protocol (TCP) and User Datagram Protocol (UDP). TCP offers reliable services and full duplex functions. UDP provides unreliable services that enhances throughput when error connection is not involved.

Internetwork Layer

This layer corresponds to the network layer of the OSI model. It is responsible for routing packets within the Internet. Gateways and routers are applicable here. The protocol at work in this layer is the TCP/IP (Transmission control protocol/ Internet protocol).

Network Access Layer

This combines the functionalities of both the physical and the datalink layers of the OSI model, it is responsible for exchange of data between a host and the network as well as receiving of data

between 2 devices on the same network. The device are identified on the network by their Internet addresses or Internet protocol (IP) addresses as shown in Figure 20.

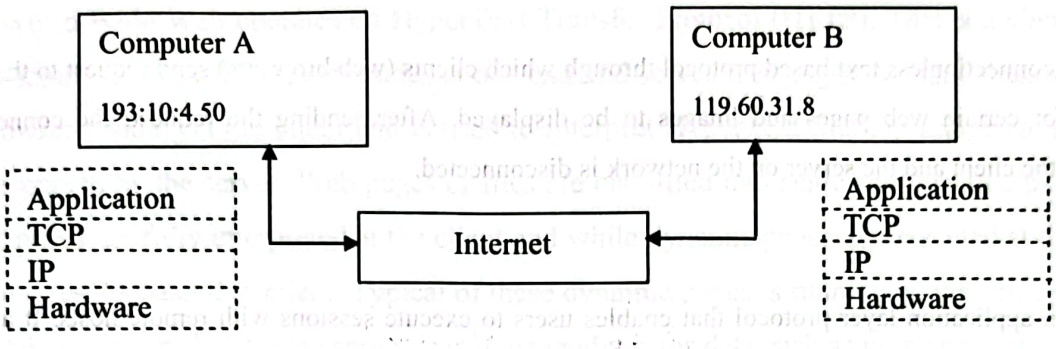


Figure 20: Interconnection of two computers on the Internet

Communication between two systems over a network can be described as follows:

1. The message that needs to be sent is written in an application on Computer A and it starts from the top using the protocol stack and moves downward.
2. If the message is large, the stack layer breaks the message into smaller chunks so that data management remains stable. The chunks of data are known as Packets.
3. The data from the application layer move towards the TCP/IP layer. The packet of the data is assigned with a port number.
4. After necessary processing at the TCP level, the packets move towards the IP layer. The IP layer provides the destination layer where the message should be received. At this level, message packets retain port number as well as IP address.
5. The hardware layer is responsible for converting alpha/numeric messages into a digital signal and sending the message through the telecommunication path.
6. Internet services provider (ISP) is also attached to the Internet, where the ISP router examines the recipient's address. The next stop of the packet is another router.
7. Eventually, the packets reach another computer. This time packets start from the bottom.
8. As the packets move upwards, the packets' data are unwrapped and certain segments are removed that was helping it to reach the destination; this includes IP address and port number. The process of wrapping and unwrapping the data is known as data encapsulation and decapsulation respectively.
9. On reaching the stack's top and Computer B, all the packets are reassembled to form the original message sent by Computer A.

Application Layer Protocols

At the application layer the following protocols are used to interpret and encrypt messages:

HTTP:

This is a connectionless text based protocol through which clients (web browsers) send request to the web servers for certain web pages and images to be displayed. After sending the request, the connection between the client and the server on the network is disconnected.

Telnet:

This is an application layer protocol that enables users to execute sessions with remote hosts. It allows login to another host at remote locations.

FTP:

This enables transfer of files between two hosts that are at remote locations to each other. It perform the basic file transfer between hosts.

SMTP (simple mail transfer protocol)

It is used for exchanging emails or basic message delivery.

SNMP (simple network management protocol)

It is used to manage the network by collecting information from a connected device on the network for management purposes.

Practical Exercise 2

1. What is open system interconnection (OSI) model? _____
   _____
2. How is it related to the Internet model? _____
   _____
3. What are the protocols at the following network layers; Application _____
   Presentation _____ Session _____
   Transport _____ Network _____
   Data link _____ Physical _____
4. What is a telnet? _____
5. Which protocol is used for receiving e-mails? _____

Unit 3: Web Architecture and Network Model

Web Architecture

The World Wide Web operates on HyperText Transfer Protocol (HTTP). This is a client-server architecture whereby the server resides at one end and serves web pages to client at another end. A browser resides on the client and is used to interpret HyperText Markup Language (HTML) codes passed by the server. Web pages or files are classified into static and dynamic pages. The static pages are fully interpreted at the client end while dynamic pages are executed at the server end and results passed to client. Typical of these dynamic pages is running queries on a database or database server. A database server is storage medium for data such as numbers, images and so on. These data are often stored in relational database structures. The web server acts as a container or storage for the web documents with extensions like .html, .asp, .php and many more. Figure 21 shows the web architecture in which clients such as laptops, PCs and workstations make HTTP requests and get HTTP responses from servers.

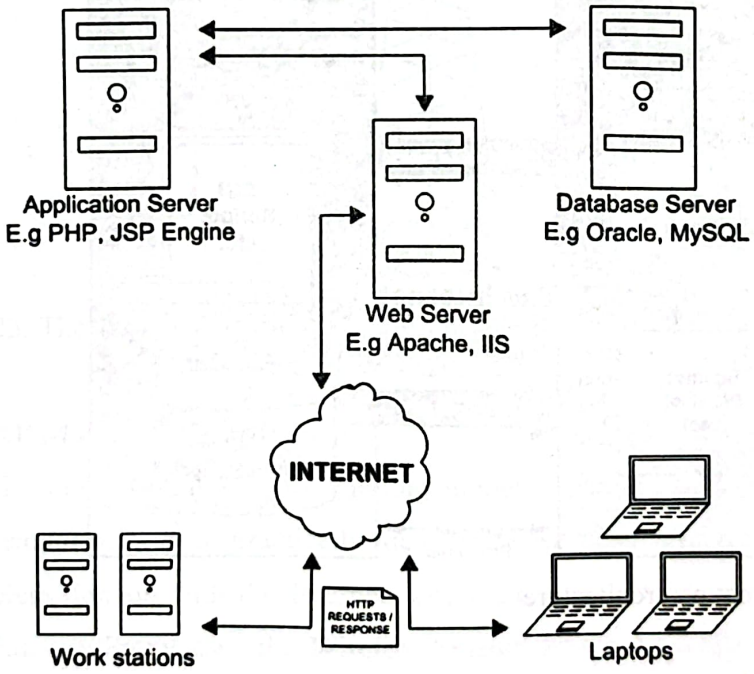


Figure 21: The Web architecture

The Web-Internet Model

The Web-Internet model makes it possible for a client to reach services on a large number of origin servers, each addressed by a unique Uniform Resource Locator (URL). The content stored on the servers is of various formats, but HTML is the predominant. HTML provides the content

developer with a means to describe the appearance of a service in a flat document structure. If more advanced features like procedural logic are needed, then scripting languages such as JavaScript or VB Script may be utilised. Figure 22 shows how a Web client request a resource stored on a web server. On the Internet standard communication protocols, like HTTP and Transmission Control Protocol/Internet Protocol (TCP/IP) are used.

The content available at the web server may be static or dynamic. Static content is produced once and not changed or updated very often; for example, a company presentation. Dynamic content is needed when the information provided by the service changes more often; for example, timetables, news, stock quotes, and account information. Technologies such as Active Server Pages (ASP), Common Gateway Interface (CGI), and Servlets allow content to be generated dynamically.

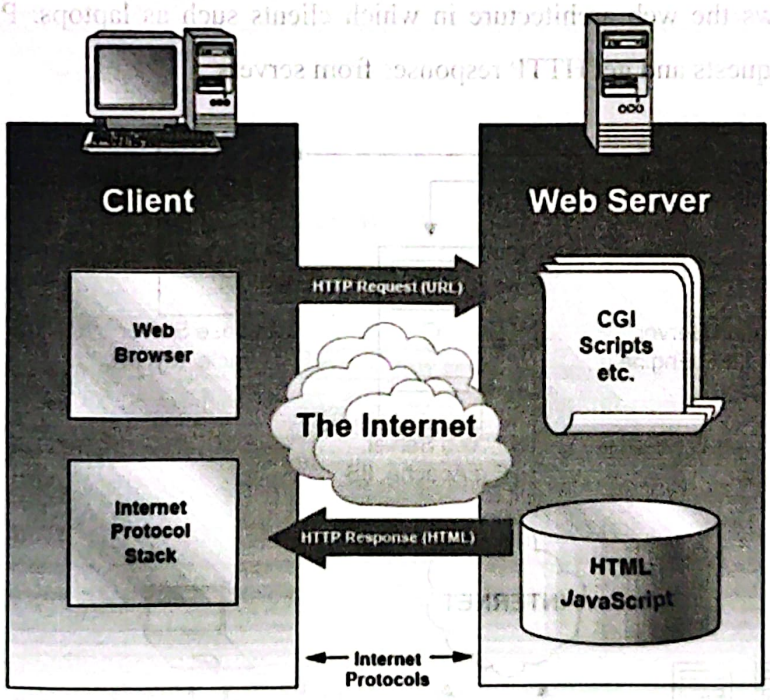


Figure 22: Web application architecture

WAP Architecture

Internet access or binary data transfer over mobile devices requires a digital communication link. Digital Communication is obtainable today with the technologies like General Radio Packet Service (GPRS) and Universal Mobile Telecommunication Service (UMTS) to mention a few. Where this link is in place, a wireless access protocol (WAP) gateway is required to interface with the existing Internet. The WAP gateway is also connected to the Transceiver Station that provides wireless connection to the mobile PCs (referred to as clients). The protocol (i.e. set of communication rules) in use is called WAP. While a browser such as Internet Explorer (IE)™ or

Mozilla Fox and Netscape Navigator is required to interpret HTML codes passed to the client in the Web Architecture. Micro browsers like Google Chrome, Microsoft Edge, Safari, Mozilla Fox, Opera, Brave, Tor, Vivaldi and Openware browsers is required to interpret Wireless Markup Language (WML) codes passed to mobile phones in WAP architecture. Figure 23 shows the WAP architecture in which data is transferred to mobile devices from server(s) via WAP gateway.

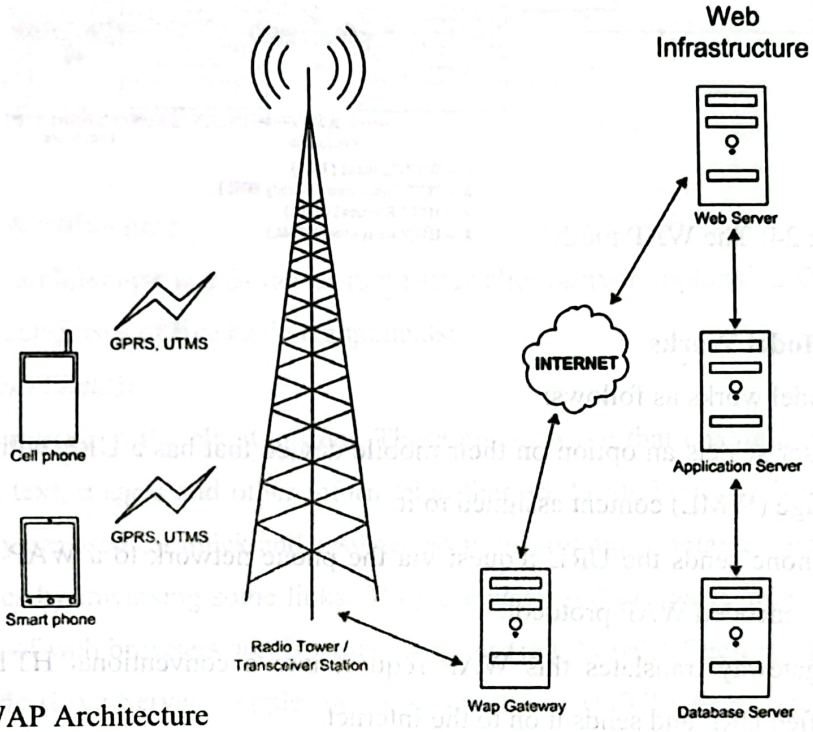


Figure 23: The WAP Architecture

The WAP Model

Figure 4 shows the WAP programming model. Without the WAP Gateway/Proxy, the two models would have been practically identical. WAP Gateway/Proxy is the entity that connects the wireless domain with the Internet. The request that is sent from the wireless client to the WAP Gateway/Proxy uses the Wireless Session Protocol (WSP). This is a binary version of HTTP. A markup language – the Wireless Markup Language (WML) is adapted to develop optimized WAP applications. In order to save valuable bandwidth in the wireless network, WML is encoded into a compact binary format. Encoding WML is one of the tasks performed by the WAP Gateway/Proxy as shown in Figure 24.

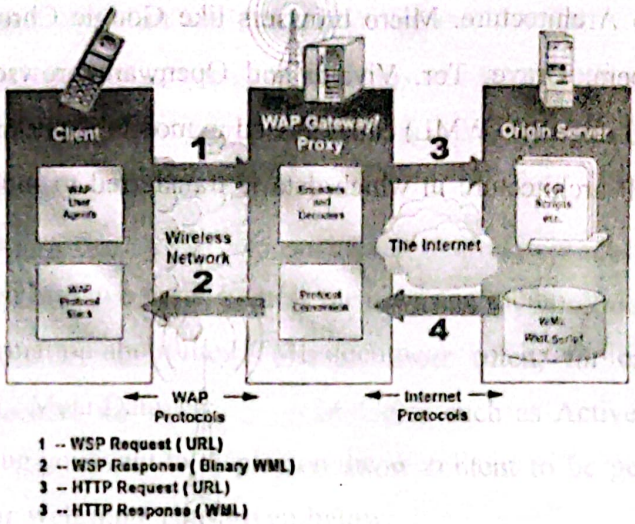


Figure 24: The WAP model

How WAP Model Works

The WAP model works as follows:

- The user selects an option on their mobile device that has a URL with Wireless Markup language (WML) content assigned to it.
- The phone sends the URL request via the phone network to a WAP gateway using the binary encoded WAP protocol.
- The gateway translates this WAP request into a conventional HTTP request for the specified URL and sends it on to the Internet.
- The appropriate Web server picks up the HTTP request.
- The server processes the request just as it would any other request. If the URL refers to a static WML file, the server delivers it. If a CGI script is requested, it is processed and the content returned as usual.
- The Web server adds the HTTP header to the WML content and returns it to the gateway.
- The WAP gateway compiles the WML into binary form.
- The gateway then sends the WML response back to the phone.
- The phone receives the WML via the WAP protocol.
- The micro-browser processes the WML and displays the content on the screen.

This is summarized in Figure 25.

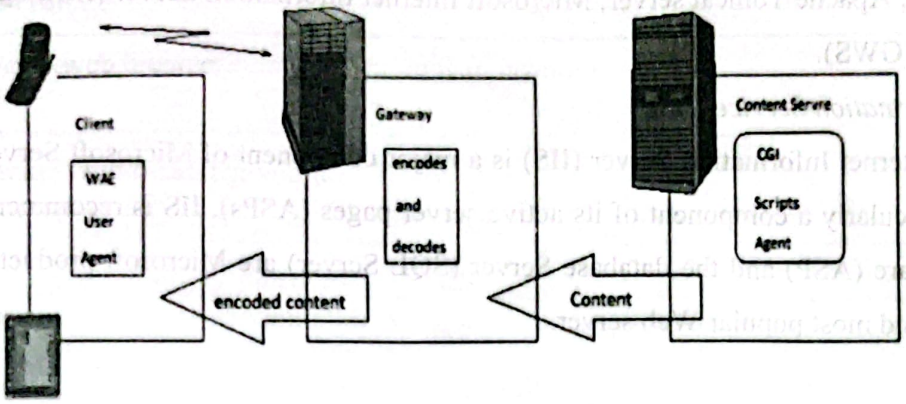


Figure 25: How the WAP model works

Web Application Architecture

A web application architecture is a 3- tier or multi-user client-server application that is run over the Internet and it comprises of five basic components:

1. ***Web Browser (Client):***

The web browser resides in the client system. These are software that enable a user to display and interact using text, images and other information that are located on a web page or a local area network. They are used for quick and easy access to information contained in the web pages or at some websites by traversing some links. They are also used to access information on web servers. Examples of web browsers are; Internet Explorer (IE), Mozilla Firefox, Chrome, Safari, Visual Studio Code (Live Server), Apple Safari, Open source HTTP Clients, Opera, etc. Web browser works by communicating with web servers using HTTP. HTTP allows web browsers to submit information to web servers as well as fetch web pages from them. The primary language of browsers is the HTML, which consists of tags that are used to describe a web page. HTML will be discussed in detailed in module 3. Most browsers also have some level of support for scripting languages like JavaScript and markup languages like extensible markup language (XML).

2. ***Web Server (Middleware):***

The web server communicates with the web browser using HTTP. All Web transactions take place on the servers. While the database server is responsible for storing the required information, the web server takes all requests from the clients, responds to the requests and serves the appropriate web pages back to the clients. Examples of web servers are: Apache

HTTP server, Apache Tomcat server, Microsoft Internet Information Server (IIS), nginx, Google Web Server (GWS).

Internet Information Service (IIS)

Microsoft Internet Information Server (IIS) is a major component of Microsoft Server operating system, particularly a component of its active server pages (ASPs). IIS is recommended if both the middleware (ASP) and the database Server (SQL Server) are Microsoft products. IIS is the world's second most popular Web server.

Apache HTTP Server

This is an open source free software that runs on; Unix, Linux, MS Windows, Mac OS X and some other platforms. Apache serves both serves both static and dynamic contents on the web in a very reliable and secure manner. It offers server-side programming language support for authentication schemes. Examples include; Hypertext pre-processor (PHP), ASP, ColdFusion, JSP and Perl.

3. ***Database Server:***

These are programs that provide database services to other systems. They are the database management systems (DBMS) that provide functionality to the database servers for storing, retrieving and manipulating data in the database or other repositories. They include; Oracle, Sybase, IBM DB2, MySQL, MS SQLServer, PostgreSQL, Open Office's base, SQLite, MS Access, SAP Sybase, Apache Derby, mSQL, MariaDB, etc.

4. ***Client-Side Programs***:

These are programs written in HTML form, JavaScript, VBScript, Flash, etc. In this course you will learn how to write client side program with HTML, XML and Java Script.

5. ***Server-Side Programs:***

Server-side programs work with other web servers to interpret requests from clients, process the requests and interact with other programs that may be needed to accomplish a task and indicate to the web server the actual page to serve the client. These are written in JavaServlets/JSP, ASP, PHP, Perl, Python, CGI, etc.

Practical Exercise 3

1. What is a web browser?

_____

2. Write their full meaning:

ASP _____ GWS _____

GPRS _____ IIS _____

CGI _____ ISP _____

SMTP _____ TCP/IP _____

UMTS _____ NIC _____

WAP _____ WML _____

WSP _____ XML _____

3. List five (5) client-side software: a._____ b._____

c. _____ d. _____ e. _____

4. List five (5) server side software: a. _____ b. _____

c. _____ d. _____ e._____

5. What is a gateway in reference to a network? _____

6. What is an application server? _____

7. Draw a block diagram of the computer network connection in the software laboratory?

[Diagram here]

Unit 4: Web Development Lifecycle

A web application development goes through a lifecycle, which provides a strategically designed methodology to achieve an elegant result. Just like the software development lifecycle, for web development it involves all the stages that go into building a website, from formulating the idea to coding, design, deployment and maintenance. It is a standards or methodological step to follow in achieving a well functional website. Web development life cycle include the following as shown in Figure 24:

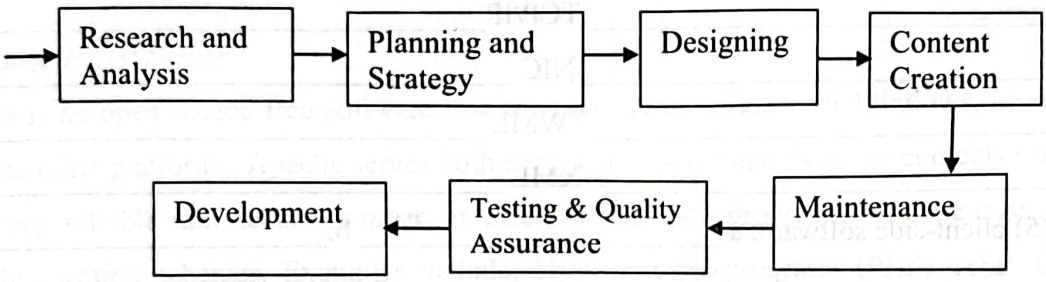


Figure 24: Web development life cycle

Research and Analysis

This phase involves gathering information about the project and the client in order to have a clear idea of what the purpose, requirements and expectations are for the web development task. During this phase, you visualize what type of people your website is going to be catering to and you take into consideration their age, gender, preferences, and needs, set goals and generate requirement elicitation documents.

Planning and Strategy

Website development plan involves strategizing all the aspects of the website including the design, technology, content, and marketing. Based on the information gathered and analyzed in the first phase, informed decisions are made about the structure and features of the website. In the Planning and Strategizing phase, a dedicated team is formed with each member having a defined role and delegated with clear tasks. Deciding on the content structure, wireframe (schematics and rough designs), choosing the technology stack and software development methodology are important decisions to take before website creation. Technology stack is a set of programming languages, web frameworks and software that are used to build any web application. Creating a **sitemap**, estimating **timelines**, defining **deliverables**, and allocation of resources are also essential parts of this phase. Afterwards we move on to the Design phase.

Designing

The design stage involves designing the website layouts and bringing in the creative user interface designers to do the task. Layouts involve designing rough sketches, which might be graphical, in order to get a feel of the design of the website. The purpose is to present an information structure enabling visual tour of the content and base features for the client. The wireframe designed in the second phase is transformed into buttons, tabs, menus, dashboards, colour schemes, typography and graphics to create a base layout of the web site.

Content Creation

Content creation involves providing relevant information about your company in an easy-to-understand, attractive manner. Adding calls-to-action, creative headlines, formatting, line editing, writing, and updating texts throughout the web development lifecycle. This phase develops the branding and marketing of the web application. This is the only way to interact with end-users and convert them into customers, so keen attention and focus is given to this phase.

Development

The development phase involves the actual building of the website. Developing the client-side and server-side of the website is accomplished in this stage. It is the most time-consuming part of the website development life cycle. Three layers of development task are identified in this phase; Client-side, Server-side and Full-stack development.

Server-side (Back-end): This is a website development task that incorporates all the processes that takes place behind the scene in a website. This involves: managing databases, servers, and logical components.

Client-side (Frontend): Task at this side mainly involves developing appealing visual representation and designing of the website. Front end developers work on creating a seamless user experience (UX) through responsive webpages using HTML, CSS, JavaScript. Good user interface will attract users to a website.

Full Stack: This involves combining the frontend and backend, which encapsulates the whole process of the website development. In this case you deal with the entire stack of tasks and technologies involved in the website development cycle.

Testing and Quality Assurance

After the website is developed, a set of rigorous tests are conducted to eliminate any bugs in the system. The quality assurance (QA) team performs repeated testing methods such as, Unit

testing, Stress testing, Integration Testing, and Load testing meticulously, checking the functionality, usability, compatibility, and performance of the web application

Maintenance

Once the QA team certify the website application okay, the web application is finally ready for deployment. Using File Transfer Protocol, the application is hosted on the web servers and is available for viewing. A continuous feedback from user interaction lets the developers know the scopes of improvement. Accordingly, the web application development life cycle is executed to make the necessary modifications. Regular maintenance and updates are absolutely performed to keep the site functioning perfectly and visibly.

Practical Exercise 4

1. What is web development life cycle in your own words? _____
2. Name the principal steps in the analysis phase _____
3. What are the 3 steps required to design a user interface and production processes? _____
4. Why is it important to obtain client feedback and adjustment web applications as appropriate?
5. What is server side development? _____
6. What is a client side web development? _____
7. Who is a full stack web developer? _____

Module 2

Hypertext Markup Language (HTML)

Unit 1: HTML Tags and Attributes

Hypertext Markup Language (HTML) is a markup language used to structure web pages. It is neither compiled nor interpreted like other programming languages but rendered directly by the browsers, displaying content without typical errors, thereby making execution smooth.

Predefined tags and attributes in HTML tells the browser how to display content. This refers to the format, style, font size, and images applicable to the display. HTML is not a case sensitive language, meaning that there is no difference in upper case and lower case, for example both 'A' and 'a' are the same.

There are generally two types of tags in HTML:

1. Paired Tags: These are tags that come in pairs. That is, they have both opening (< >) and closing (</ >) tags.
2. Empty Tags: These are tags that do not require to be closed.

An example of HTML tag is the (<b>) tag in HTML, which tells the browser to bold the text inside it.

Practical Exercise 1

Using any HTML tag of your choice, decide a display to your sample web page. Write your code in the box below:

Tags and attributes:

HTML tags are structural components enclosed in angle brackets. They're opened and closed with a forward slash (e.g., <h1></h1>). Some tags are self-closing, while others allow attributes like width, height, and controls for defining properties or storing metadata.

Unit 2: Structure of an HTML page

An HTML Document is mainly divided into two parts, namely the head and the body:

- **HEAD:** This contains the information about the HTML document. For Example, the Title of the page, version of HTML, Meta Data, etc.
- **BODY:** This contains everything you want to display on the Web Page.

The structure of HTML document or web page is as given below:

```
<!DOCTYPE html>
<html>

<head>
    <title>Page Title</title>
</head>

<body>
    <h2>Heading Content</h2>
    <p>Paragraph Content</p>
</body>

</html>
```

HTML Document Structure

The basic structure of HTML with necessary code for every webpage is as given below:

```
<html>
<!-- Defines types of documents : Html 5.O  -->
<!DOCTYPE html>
<!-- Defines types of documents : Html 5.O -->
<html lang="en">
  <!-- DEfines languages of content : English -->

 <head>
   <!-- Information about website and creator -->
   <meta charset="UTF-8" />
   <meta
     http-equiv="X-UA-Compatible"
```

```
      content="IE=edge"
    />
    <!-- Defines the compatibility of version with browser -->
    <meta
      name="viewport"
      content="width=device-width,
          initial-scale=1.0"
    />
    <!-- for make website responsive -->
    <meta name="author" content="Mr.X" />
    <meta
      name="Linkedin profile"
      content="WWW.linkedin.com/Mr.X_123"
    />
    <!-- To give information about author or owner -->
    <meta
      name="description "
      content="A better place to learn computer science"
    />
    <!-- to explain about website in few words -->
    <title>Web Programming</title>
    <!-- Name of website or content to display -->
  </head>

  <body>
    <!-- Main content of website -->
    <h1>Web Programming</h1>
    <p>A computer science course portal</p>
  </body>
</html>
```

The complete explanation of each of the tags used in the above piece of HTML code is given below:

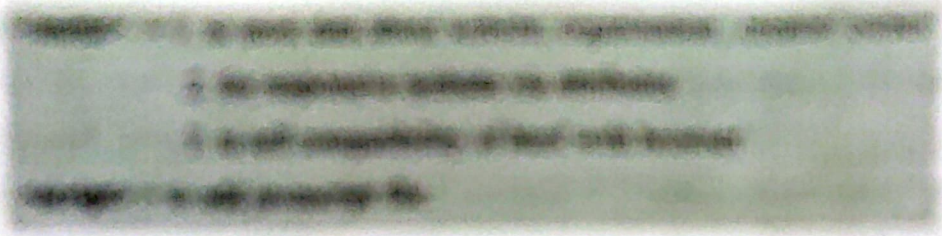
- **DOCTYPE Declaration (<!DOCTYPE html>):** Specifies the HTML version; typically, it indicates HTML5.
- **<html>:** Root element encompassing the entire HTML document structure.
- Parent to <head> and <body> tags.
- Specifies language with the "lang" attribute (e.g., lang="en" for English).
- **<head>:** Container for metadata, title, CSS, scripts, etc.
- Content isn't directly displayed but serves informational and structural purposes.

Note: for better understanding refer above code of html.

<title> = to store website name or content to be displayed.

<link> = To add/ link css (cascading style sheet) file.

[illegible]



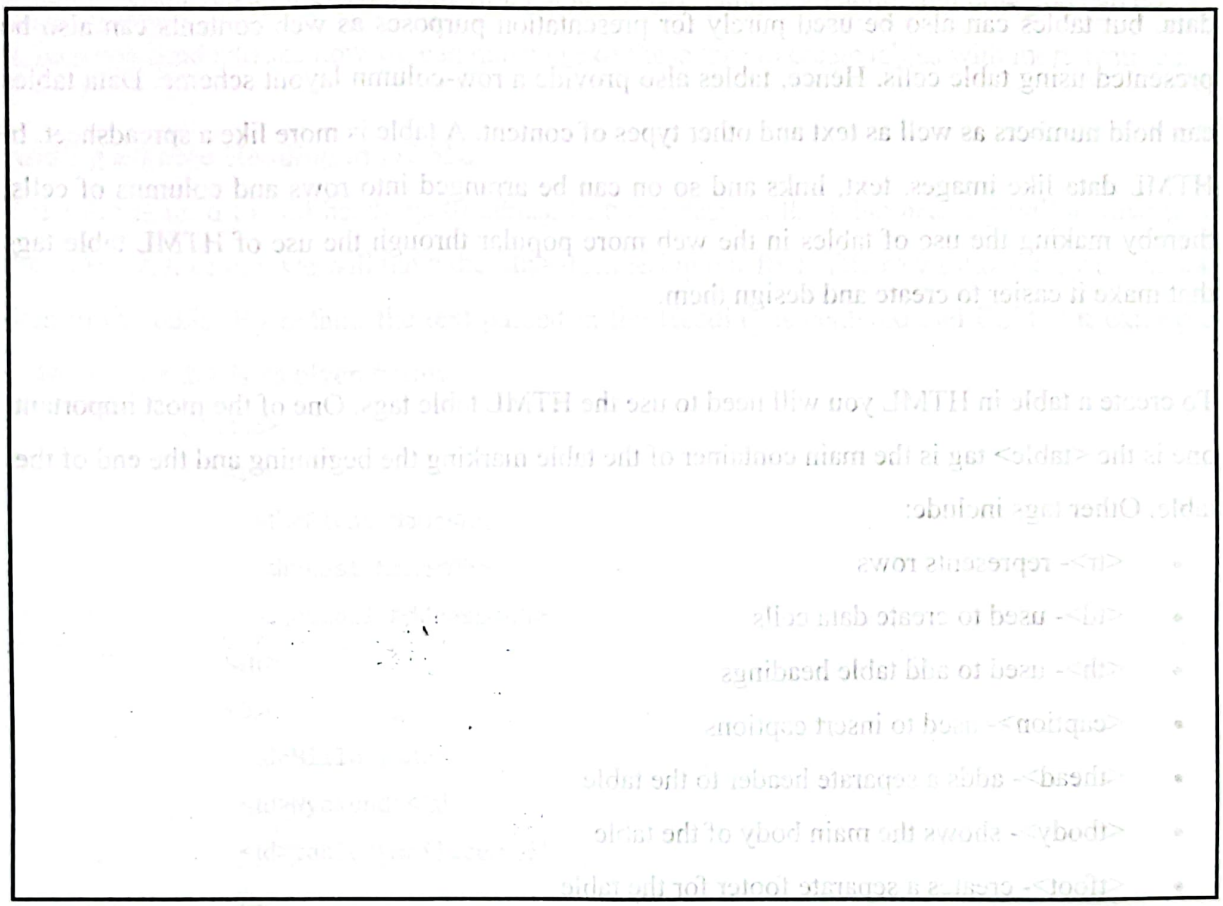
4. [illegible]

[illegible]

1. [illegible]
2. [illegible]
3. [illegible]
4. [illegible]
5. [illegible]
6. [illegible]
7. [illegible]
8. [illegible]
9. [illegible]

Practical Exercise 2

Using the basic structure explained above, Design a web page for this course showing necessary tags and elements as desired. Attach the print-out of the displayed web page from your code. Write your code in the box provided here (Attach paper if necessary).



Unit 3: HTML Lists and Tables

The HTML List and Table tags are enumerated here:

HTML List Tags: <ul>, <ol>, <li>, <dl>, <dt> and <dd>

HTML Table Tags: table, tr, td, th, tbody, thead, tfoot, col, colgroup and caption

HTML Form Tags: form, input, textarea, select, option, optgroup, button, label, fieldset and legend

HTML Scripting Tags: script and noscript

A Table in HTML

A table is a group of cells organized in a two-dimensional structure with rows and columns. A table is a representation of data arranged in rows and columns. It is one of the organizational construct required to organize a web page's content. A table's cells is normally known to hold data, but tables can also be used purely for presentation purposes as web contents can also be presented using table cells. Hence, tables also provide a row-column layout scheme. Data tables can hold numbers as well as text and other types of content. A table is more like a spreadsheet. In HTML data like images, text, links and so on can be arranged into rows and columns of cells, thereby making the use of tables in the web more popular through the use of HTML table tags that make it easier to create and design them.

To create a table in HTML you will need to use the HTML table tags. One of the most important one is the <table> tag is the main container of the table marking the beginning and the end of the table. Other tags include:

- <tr>- represents rows
- <td>- used to create data cells
- <th>- used to add table headings
- <caption>- used to insert captions
- <thead>- adds a separate header to the table
- <tbody>- shows the main body of the table
- <tfoot>- creates a separate footer for the table

The HTML Table Syntax is as given below:

```
<table>
 <tr>
  <td>Cell 1</td>
  <td>Cell 2</td>
  <td>Cell 3</td>
 </tr>
 <tr>
  <td>Cell 4</td>
  <td>Cell 5</td>
  <td>Cell 6</td>
 </tr>
</table>
```

This code will display a table with six cells as given below:

Cell 1	Cell 2	Cell 3
Cell 4	Cell 5	Cell 6

Now that you have an understanding of what an HTML table is all about and how you can create it, let's go ahead and see how we can make use of these tags to create tables with more features.

Adding a Table Heading in HTML

The <th> is used to add headings to tables. In basic designs the table heading will always take the top row, meaning we will have the <th> declared in our first table row followed by the actual data in the table. By default the text passed in the Heading is centered and Bold. **An example with use of <th> is as given below:**

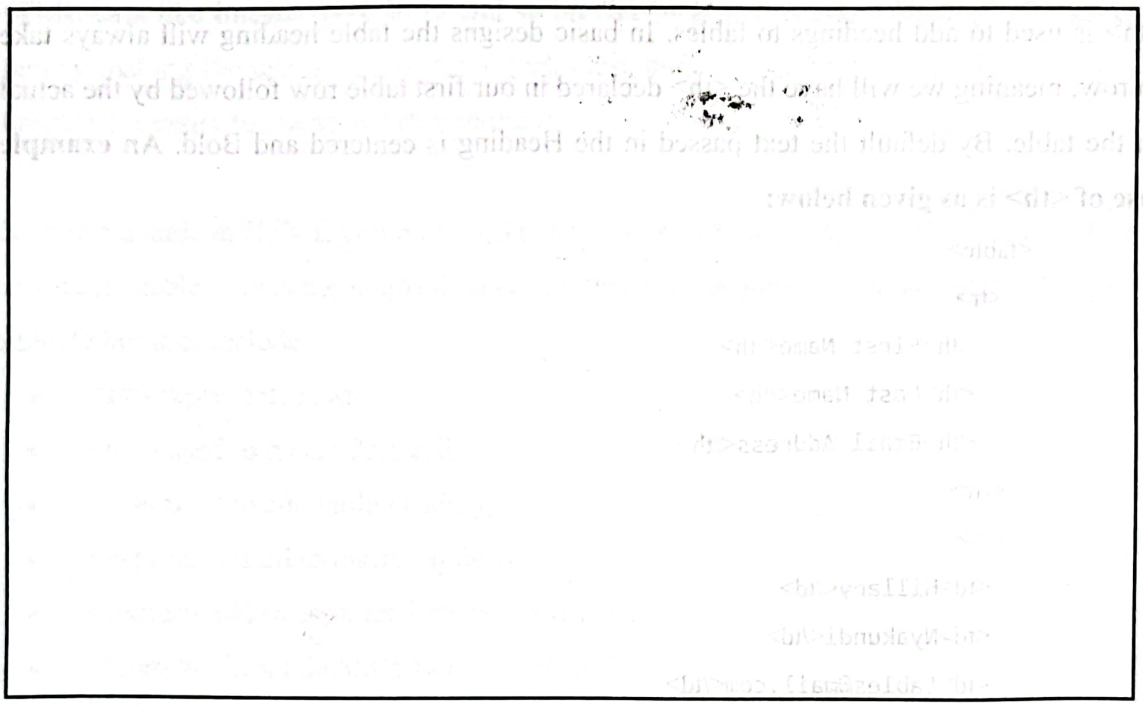
```
<table>
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Email Address</th>
  </tr>
  <tr>
   <td>Hillary</td>
   <td>Nyakundi</td>
   <td>tables@mail.com</td>
  </tr>
  <tr>
    <td>Larry</td>
    <td>Mark</td>
    <td>developer@mail.com</td>
  </tr>
</table>
```

FIRST NAME	LAST NAME	EMAIL ADDRESS
Hillary	Nyakundi	tables@mail.com
Larry	Mark	developer@mail.com

From the example above, we are able to tell what column contains which information. This is made possible with the use of <th> tag.

Practical Exercise 3

Write your HTML code to present the records of students offering CSC 121 such as Name, Registration No., Date of Birth, State of Origin, etc. (Attach printout of your displayed web page showing the table).



Adding a Caption to a Table

The main use of adding a caption to table is to provide a description about the data represented in the table. By default, the caption will always be centered, otherwise it can either be placed at the top of the table or the bottom. To insert a caption into a table, the <caption> tag is used and the syntax below can be followed.

```
<caption></caption>
<tr> </tr>
</table>
```

An example with use of<caption>

```
<table>
 <caption>Free Coding Resources</caption>
 <tr>
```

```
    <th>Sites</th>
    <th>Youtube Channels</th>
    <th>Mobile Apps</th>
  </tr>
  <tr>
    <td>Freecode Camp</td>
    <td>Freecode Camp</td>
    <td>Enki</td>
  </tr>
  <tr>
    <td>W3 Schools</td>
    <td>Academind</td>
    <td>Programming Hero</td>
  </tr>
  <tr>
    <td>Khan Academy</td>
    <td>The Coding Train</td>
    <td>Solo learn</td>
  </tr>
</table>
```

The resulting table below is obtained:

SITES	YOUTUBE CHANNELS	MOBILE APPS
Freecode Camp	Freecode Camp	Enki
W3Schools	Academind	Programming Hero
Khan Academy	The Coding Train	Solo learn

Free Coding Resources

How to Use the Scope Attribute in HTML Tables

The scope attribute is used to define whether a specific header is intended for either a column, row, or a group of both. I know the definition might be challenging to understand but hold on – with the help of an example you will better understand it.

The main purpose of the scope element is to show the target data so that the user does not have to rely on assumptions. The attribute is declared in the header cell <th>, and it takes the values col, row, colgroup and rowgroup.

The values col and row indicated that the header cell is providing information for either the rows or columns respectively.

Scope Syntax

```
<table>
 <tr>
  <th scope="value">
 </tr>
</table>
```

An Example with use of <scope> is hereby given:

```
<table>
 <tr>
  <th></th>
  <th scope="col">Semester</th>
  <th scope="col">Grade</th>
 </tr>

 <tr>
  <td>1</td>
  <td>Jan - April</td>
  <td>Credit</td>
 </tr>

 <tr>
  <td>2</td>
  <td>May - August</td>
  <td>Pass</td>
 </tr>

 <tr>
  <td>2</td>
  <td>September - December</td>
  <td>Distinction</td>
 </tr>
</table>
<table>
      <th>NAME</th><th>SUBJECTS</th><th>MARKS</th>
       </tr>
       <tr>
        <td rowspan = "2">Hillary</td>
        <td>Advanced Web</td>
        <td>75</td>
       </tr>
       <tr>
        <td>Operating System</td>
        <td>60</td>
```

```
</tr>
  <tr>
 <td rowspan = "2">Larry</td>
 <td>Advanced Web</td>
 <td>80</td>
</tr>
<tr>
 <td>Operating System</td>
 <td>75</td>
</tr>
<tr>
  <td colspan="3">Total Average: 72.5</td>
 </tr>
</table>
```

Resulting Display:

NAME	SUBJECT	MARKS
	Advanced Web	75
Hillary	Operating System	60
	Advanced Web	80
Larry	Operating System	75
	Total Average: 72.5	

In the example above, we have a cell spanning of 2 cells in the rows and 3 cells in the column as indicated. We have managed to apply the span both vertically and horizontally. *When using the attributes colspan and rowspan, make sure to declare the values assigned correctly to avoid overlapping of cells.*

Adding a Table Header, Body and Footer in HTML Tables

Just like how a website or any other document has three main sections – the header, body, and footer – so does a table. In a table they are divided by using attributes namely:

- <thead>- provides a separate header for the table
- <tbody>- contains main content of the table
- <tfoot>- creates a separate footer for the table

An Example with use of<thead>,<tbody> and<tfoot>

```
<table>
 <thead>
  <tr>
   <th colspan="2">October</th>
   <th colspan="2">November</th>
  </tr>
 </thead>

 <tbody>
  <tr>
   <td>Sales</td>
   <td>Profit</td>
   <td>Sales</td>
   <td>Profit</td>
  </tr>
  <tr>
   <td>₦200,00</td>
   <td>₦ 50,00</td>
   <td>₦ 300,000</td>
   <td>₦ 70,000</td>
  </tr>
 </tbody>

 <tfoot>
  <tr>
   <th colspan= "4">November was more productive</th>
  </tr>
 </tfoot>
</table>
```

Resultant Display

OCTOBER		NOVEMBER	
Sales	Profit	Sales	Profit
₦200,00	₦50,00	₦300,000	₦70,000
NOVEMBER WAS MORE PRODUCTIVE			

From the above example, the header is represented by the name of the months, the part with the figures of both sales and profit represents the table body, and finally the part with the statement represents the footer of our table. Another important thing to note is that a table can have more than one body part. In a scenario like this each body groups the rows that are related together.

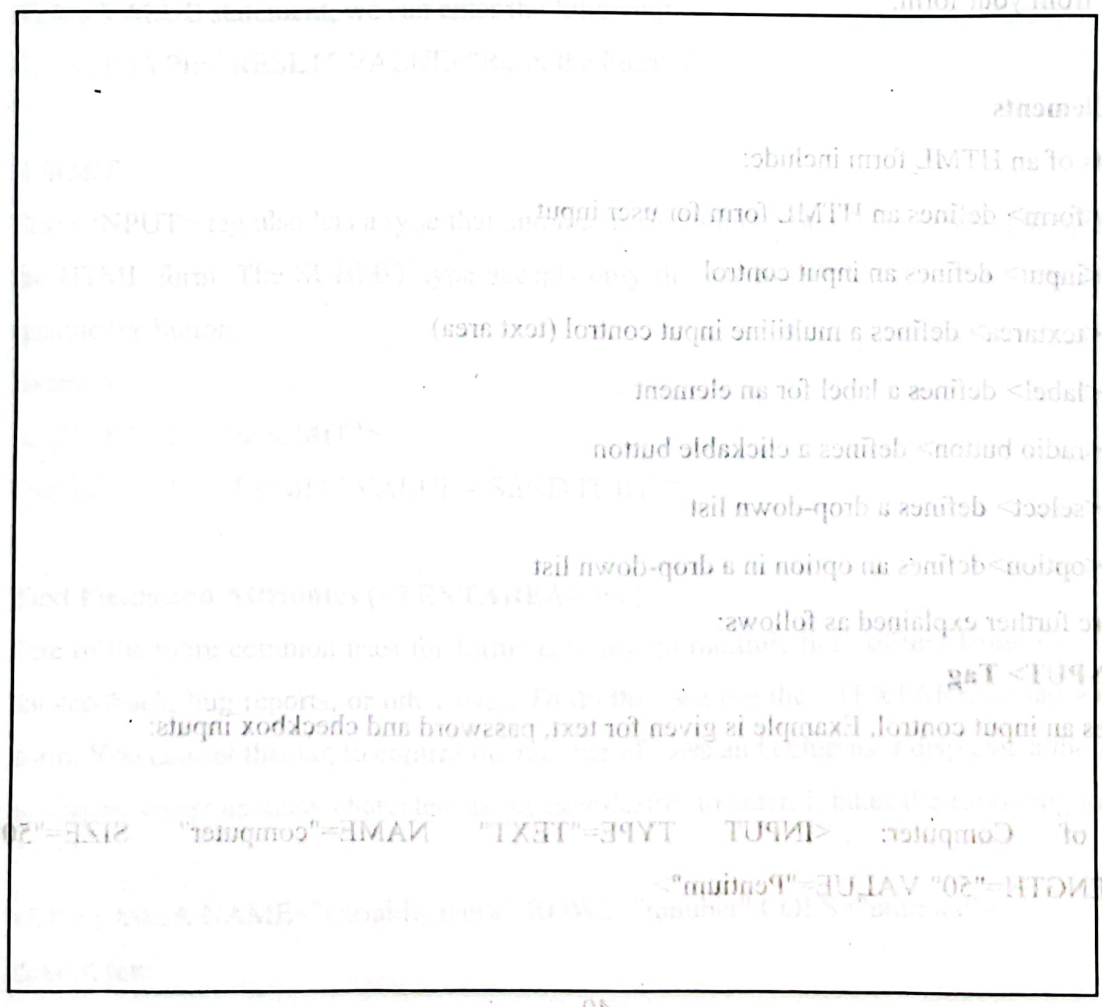
When to Use a Table

There are many situations where tables might come in handy when developing your projects:

- You can use tables when you want to compare and contrast data with shared characteristics like the differences between A and B or scores of team X to those of Y.
- You can also use one if you want to give an overview of numerical data. A good example of this is when you are trying to represent marks of students or even the scores of teams like the EPL table.
- And a table can help readers quickly find specific information laid out in a clear way. For example if you are going through a long list of name a table can be used to sub divide the list which make it easy for readers.

Practical Exercise 4

Update your table in practical exercise 3 with the additional features that you have learnt since then. Put down your code in the space provided. Attach your resulting table and any extra sheets where applicable.



Unit 4: HTML Form processing, Creating HTML documents with Tables, Forms and multimedia

Forms are used to collect user input. This can be added to an HTML document.

Creating the HTML Form

In an HTML document, forms are set between the <FORM> container tags. The form container works as follows:

<FORM METHOD="how_to_send" ACTION="URL of script">
...form data...
</FORM>

Notice that the <FORM> tag takes two attributes: METHOD and ACTION. The METHOD attribute accepts either POST or GET as its value.
The second attribute is ACTION, which simply accepts the URL for the script that will process the data from your form.

Form Elements

Elements of an HTML form include:

- <form> defines an HTML form for user input
- <input> defines an input control
- <textarea> defines a multiline input control (text area)
- <label> defines a label for an element
- <radio button> defines a clickable button
- <select> defines a drop-down list
- <option>defines an option in a drop-down list

These are further explained as follows:

The <INPUT> Tag

It defines an input control. Example is given for text, password and checkbox inputs:

Text

Type of Computer: <INPUT TYPE="TEXT" NAME="computer" SIZE="50" MAXLENGTH="50" VALUE="Pentium">

Password

Enter Password: <INPUT TYPE="PASSWORD" NAME="password" SIZE="25" MAXLENGTH="25">

Checkbox

Type of computer (s) you own :< BR>

<INPUT TYPE="CHECKBOX" NAME="Pentium" CHECKED> Pentium

<INPUT TYPE="CHECKBOX" NAME="486"> 486-Series PC

<INPUT TYPE="CHECKBOX" NAME="Macintosh"> Macintosh

RESET

This <INPUT> tag has built into it the ability to clear an HTML form

<INPUT TYPE="RESET">

With a VALUE statement, we can enter the following:

<INPUT TYPE="RESET" VALUE="Reset the Form">

SUBMIT

This <INPUT> tag also has a type that automatically submits the data that has been entered into the HTML form. The SUBMIT type accepts only the attribute VALUE, which can be used to rename the button.

Example:

<INPUT TYPE="SUBMIT">

INPUT TYPE="SUBMIT" VALUE="SEND IT IN!">

Text Fields and Attributes (<TEXTAREA> tag)

One of the more common uses for forms is to accept multiple lines of text from a user, perhaps for feedback, bug reports, or other uses. To do this, we use the <TEXTAREA> tag within your form. You can set this tag to control the number of rows and columns it displays, although it will generally accept as many characters as the user desires to enter. It takes the following form:

<TEXTAREA NAME="variable_name" ROWS="number" COLS="number">

default text

</TEXTAREA>
<FORM>
<TEXTAREA NAME="comments" ROWS="4" COLS="40">
Enter comments about this Web site
Good or Bad.
</TEXTAREA>
</FORM>

Radio Button

Unlike checkbox, however, Radio button is also designed to accept only one response from among its options. Radio button uses the same attributes and basic format as Checkbox.

Radio button requires that you we use the VALUE attribute, and that the NAME attribute be the same for all of <INPUT> tags that are intended for the same group. VALUE, on the other hand, should be different for each choice. For instance, look at the following example:

Choose the computer type you use most often :< BR>
<INPUT TYPE="RADIO" NAME="Computer" VALUE="P" CHECKED> Pentium
<INPUT TYPE="RADIO" NAME="Computer" VALUE="4"> 486-Series PC
<INPUT TYPE="RADIO" NAME="Computer" VALUE="M"> Macintosh
<INPUT TYPE="RADIO" NAME="Computer" VALUE="O"> Other

Creating an HTML Forms with Tables

To create a form table with input fields. For example for, surname, other names, email, phone, and address. An example of how we can structure the form table is as follows:

Each <tr> represents a row in the table, and within each row, we have two <td> tags, one for the label and one for the input field. The *for* attribute of the <label> tag is used to associate it with the corresponding input field using the id attribute.

Example:

<html>
<head>
<body>
<table>
<tr>

```
<td><label for="surname">Surname:</label></td>
 <td><input type="text" id="surname" name="surname"></td>
</tr>
<tr>
 <td><label for="othernames">Other Names:</label></td>
 <td><input type="text" id="othernames" name="othernames"></td>
</tr>
<tr>
 <td><label for="email">Email:</label></td>
 <td><input type="email" id="email" name="email"></td>
</tr>
<tr>
 <td><label for="phone">Phone:</label></td>
 <td><input type="tel" id="phone" name="phone"></td>
</tr>
<tr>
 <td><label for="address">Address:</label></td>
 <td><input type="text" id="address" name="address"></td>
</tr>
</table>
  </body>
</html>
```

The resultant display:

Surname:
Other Names:
Email:
Phone:
Address:

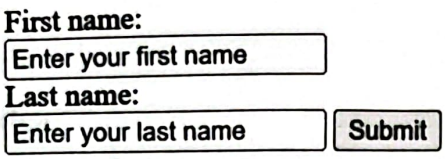
Example:

To create a text input, we use the <input> element.

```
<form action="/Test.html" method="POST">
  <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname" placeholder="Enter your first name"><br>
```

```
  <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname" placeholder="Enter your last name">
  <input type="submit" value="Submit">
</form>
```

Resultant display:

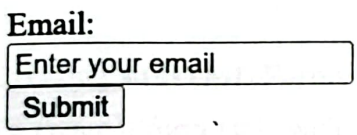


Example:

To create e-mail input, we use the type "email".

```
<form>
  <label for="email" id="email-lable">Email:</label><br>
    <input type="email" id="email" name="email" placeholder="Enter your email"><br>
  <input type="submit" value="Submit">
</form>
```

The resultant display:



Example:

To make a submit button in a form, we use <input> with the type "submit". If there are no value attributes, the default text, Submit, shows up on the button.

```
<form>
 <input type="submit" value="Send">
</form>
```

Example:

To make a Password input:

```
<form>
```

```
<label for="user-password">Password: </label>
<input type="password" id="user-password" name="user-password">
</form>
```

Password:

Example:

For number inputs:

```
<form>
 <label for="years"> Years of experience: </label>
 <input id="years" name="years" type="number" step="1">
</form>
```

Years of experience: 2

Example:

For range input, we create a slider.

```
<form>
 <label for="volume"> Volume Control</label>
 <input id="volume" name="volume" type="range" min="0" max="100" step="1">
</form>
```

Volume Control

Example:

For checkbox input:

```
<form>
 <p>Choose your ice cream flavor:</p>
<input id="vanilla" name="topping" type="checkbox" value="vanilla">
 <label for="vanilla">Vanilla</label>
 <br>
 <input id="chocolate" name="topping" type="checkbox" value="chocolate">
 <label for="chocolate">Chocolate</label>
 <br>
 <input id="strawberry" name="topping" type="checkbox" value="strawberry">
 <label for="strawberry">Strawberry</label>
```

```
</form>
```

Choose your ice cream flavor:

- ☐ Vanilla
- ☐ Chocolate
- ☐ Strawberry

Example:

For Radio button input:

```
<form>
  <p>What is sum of 1 + 1?</p>
  <input type="radio" id="two" name="answer" value="2">
  <label for="two">2</label>
  <br>
  <input type="radio" id="eleven" name="answer" value="11">
  <label for="eleven">11</label>
</form>
```

What is sum of 1 + 1?

- ◉ 2
- ○ 11

Example:

Form for drop-down lists:

```
<form>
  <label for="lunch">What's for lunch?</label>
  <select id="fruits" name="fruits">
    <option value="mango">Mango</option>
    <option value="apple">Apple</option>
    <option value="banana">Banana</option>
  </select>
</form>
```

What's for lunch? Apple

Example:

Form for datalist input

```
<form>
 <label for="lunch">What's your favorite fruit?</label>
 <input type="text" list="fruits" id="fruit" name="fruit">
 <datalist id="fruits">
  <option value="mango">Mango</option>
  <option value="apple">Apple</option>
  <option value="banana">Banana</option>
 </datalist>
</form>
```

What's your favorite fruit? Guava, Orange and Apples

Example:

Form for Text area.

```
<form>
 <label for="comment">Comment: </label>
 <br>
 <textarea id="comment" name="comment" rows="5" cols="30" placeholder="Adding text">
 </textarea>
</form>
```

Comment:

This is a text area for entering text in a form

Adding Graphics to Web Pages

We can choose either of two file types: GIF and JPEG. GIF (CompuServe Graphics Interchange Format) is the more popular among Web browsers, but JPEG (Joint Photographic Experts Group) is also popular and widely used.

To add graphics, you use an empty tag called the <IMG> (image) tag, which you insert into the body section of the HTML document as follows:

```
<IMG SRC="image URL">
```

Or

<IMG SRC="path/filename">

SRC accepts the name of the file that you want to display, and image URL (or path/filename) is the absolute (full URL) or relative path (for a local file or a file in the current directory) to the image.

Example:

<IMG SRC="C:\Dept of Software Engineering\Uniuyologo.jpg" ALT="Department of Software Engineering">

<h1>Welcome to Department of Software Engineering Website</h1>

<h2><IMG SRC="C:/vicpix/vicpic.jpg" ALT="Photo of the Head of Department"</h2>

<h2><ALIGN=MIDDLE>I am the HOD of the Department of Software Engineering, Faculty of Computing</h2>

</body>

</html>

Adding Hypertext and Creating Links

Using the <A> Tag

<A HREF="URL">Text describing link</A>

<A HREF="http://www.uniuyo.edu.ng/fc/se">Department of Software Engineering Information</A>

Aside from creating hypertext links to documents on your local computer or elsewhere on the Internet, you can create links to other parts of the same document in which the link appears. following this format:

<A HREF="#section_name">Link to another section of this document</A>

<A NAME="section_name">Beginning of new section</A>

Hyperlinks for E-Mail Messages

<A HREF="mailto:webmaster@uniuyo.edu.ng">Send me e-mail</A>

Creating Graphical Links

We can create graphical links as follows:

<A HREF="http://www.uniuyo.edu.ng/"> <IMG SRC="uniuyologo.gif" ALT="University of Uyo"></A>

Creating a Graphical Menu Bar

We can create graphical menu bar as follows:

<BODY>
<A HREF="http://www.uniuyo.edu.ng/index.html"><IMG SRC="home_button.gif"
ALT="Back to Home"></A>
<A HREF="http://www.uniuyo.edu.ng/Faculties.html"><IMG SRC="fac_button.gif"
ALT="To Faculties"></A>
<A HREF="http://www. uniuyo.edu.ng /about.html"><IMG SRC="about_button.gif"
ALT="To About University of Uyo"></A>
<A HREF="http://www. uniuyo.edu.ng /service.html"><IMG SRC="serv_button.gif"
ALT="To Services"></A>
</BODY>

Practical Exercise 5

Create an Order Form. Format the given form and perform the following actions:

a. Insert the missing document tags and give the web page a title

b. Make good use of the following to present a well formatted web page

   i. Container tags

   ii. Physical tags

   iii. Empty tags

c. Change the value of the submit button to SUBMIT

d. Change the value of the reset button to RESET THE FORM

e. Assign to the various forms maximum length attribute as you deem fit

f. Make the Internet/Web the default checked item

g. Identify the comment statement and eliminate.

Paste the resulting line of codes here or attach additional paper.

Module 3

Cascading Style Sheets (CSS)

Unit 1: Basics of Cascading Style Sheet (CSS)

This tool is used for the presentation of the Web page. Depending on the content and purpose of your website, the browser's default rendering of document could be perfectly adequate. Otherwise, the presentation is changed with Cascading Style Sheet (CSS). Therefore, to change the appearance of the text elements and the page background, simple style sheet rules apply.

Recall that the structural layer of the Webpage is created using HTML. This layer is where the presentation layer is applied on Web page. CSS is used for Web page presentation where elements that accurately describe the meaning of the content is chosen. An understanding of the document's structure and the relationships between elements is central to your work as a style sheet author.

How style sheets work

1. Start with a document that has been marked up in HTML.
2. Write style rules for how you would like certain elements to appear on the Webpage.
3. Attach the style rules to the document. When the browser displays the document, it follows your rules for rendering elements (unless the user has applied some mandatory styles)

Writing the Rules

A style sheet is made up of one or more style instructions (called style rules) that describe how an element or group of elements should be displayed. Each rule selects an element and declares its appearance. The two main sections of a rule are **the selector** that identifies the element or elements to be affected, and **the declaration** that provides the rendering instructions. The declaration, in turn, is made up of a property (such as: color) and its value (e.g. green), separated by a colon and a space. One or more declarations are placed inside curly brackets.

The general form of declaration rules is as follows,

```
selector { property: value; }
while the declaration block is as follows:
 selector {
        property1: value1;
        property2: value2;
        property3: value3;
    }
```

Selectors

where *h1* and *p* elements are used as selectors. This is called an element type selector, and it is the most basic type of selector. The properties defined for each rule will apply to every *h1* and *p* element in the document, respectively.

ID selector selects an element based on the value of an element's id attribute. It is indicated with the # symbol.

For example, the selector *#recipe* targets an element with *id="recipe"*.

Declarations

The declaration is made up of a property/value pair. There can be more than one declaration in a single rule.

Properties

The heart of style sheets lies in the collection of standard properties that can be applied to selected elements. The complete CSS specification defines dozens of properties for everything from text indents to how table headers should be read aloud.

The following example contains two rules:

i) h1 { color: green; }

The rule in (i) makes all the *h1* elements in the document green;

ii) p { font-size: large; font-family: sans-serif; }

The second CSS rule specifies the paragraphs in a large, sans-serif font. Sans-serif fonts do not have a little slab (a serif) at the ends of strokes and tend to look more sleek and modern.

From these rules, *h1* and *p* elements are used as selectors. This is called an element type selector, and it is the most basic type of selector. The properties defined for each rule will apply to every h1 and p element in the document, respectively.

Practice Exercise 1

1. Define two types of CSS rules, showing any selector and declaration elements each.

_____

_____

_____

_____

2. Write complete code snippets involving embedding the style rules in an HTML document [Attach additional sheet of paper]

Unit 2:Adding style to HTML

To add style to HTML, the following steps are required:

i) Open index.html.

```
<head>
<meta charset="utf-8">
<title>UNIVERSITY OF UYO</title>
<style>

</style>
</head>
```

ii) Place the style element inside the document head as shown in the HTML code in step i.

iii) Start by adding the style element to the document by typing the following style rules within the style element:

```
<style>
body {
background-color: #faf2e4;
margin: 0 10%;
font-family: sans-serif;
}
h1 {
text-align: center;
font-family: serif;
font-weight: normal;
text-transform: uppercase;
border-bottom: 1px solid #57b1dc;
margin-top: 30px;
}
h2 {
color: #d1633c;
font-size: 1em;
}
</style>
```

(iv) Save the file and take a look at it in the browser

To add style to HTML content, any of the **external, embedded or inline style** sheets can be used for the purpose.

External style sheets

An external style sheet is a separate, text-only document that contains a number of style rules. It must be named with the .css suffix. The .css document is then linked to (via the link element) or

imported (via an @import rule in a style sheet) into one or more HTML documents. In this way, all the files in a website may share the same style sheet. This is the most powerful and preferred method for attaching style sheets to content.

Embedded style sheets

It is placed in a document via the style element, and its rules apply only to that document. The style element must be placed in the head of the document. This example also includes a comment (see the "Comments in Style Sheets" sidebar).

```
<head>
<title>Required document title here</title>
<style>
/* style rules go here */
</style>
</head>
```

Inline styles

Properties and values can be applied to a single element by using the style attribute in the element itself. Example:

```
<h1 style="color: red">Introduction</h1>
```

To add multiple properties, just separate them with semicolons, like this:

```
<h1 style="color: red; margin-top: 2em">Introduction</h1>
```

Inline styles apply only to the particular element in which they appear. It should be avoided, unless it is absolutely necessary to override styles from an embedded or external style sheet. Inline styles are problematic in that they intersperse presentation information into the structural markup. They also make it more difficult to make changes because every style attribute must be hunted down in the source.

Inheritance

Inheritance provides a mechanism for styling elements that do not have any explicit styles rules of their own. There are things to consider for accurate implementation of inheritance, namely:

Document structure: The understanding of the document structure is important in the presentation of a Web page.

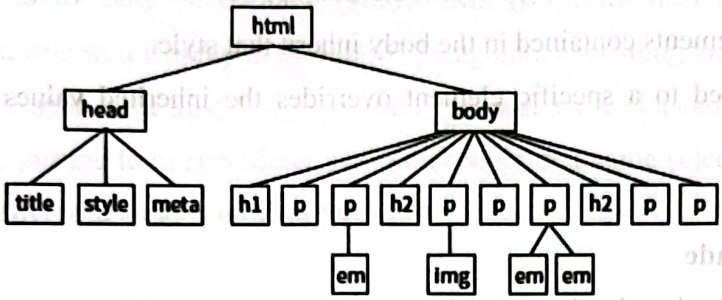


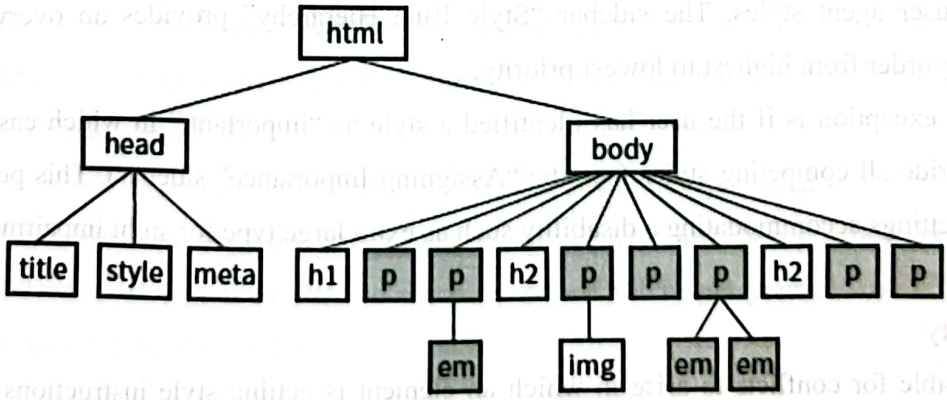
Figure 1: A typical document structure

Parents and children

The document tree becomes a family tree when it comes to referring to the relationship between elements. All the elements contained within a given element are said to be its descendants. For example, the ***h1, h2, p, em***, and ***img*** elements in the document are all descendants of the body element.

Pass it on

When a font-related style rule written using the ***p*** element as a selector, the rule applies to all of the paragraphs in the document as well as the inline text elements they contain. In general, properties related to the styling of text—***font size, color, style,*** and the like—are passed down. Properties such as ***borders, margins, backgrounds,*** and so on that affect the boxed area around the element tend not to be passed down. For example, if you put a border around a paragraph, you would not want a border around every inline element (such as em, strong, or a) it contains as well.




```
p {font-size: large; font-family: sans serif;}
```

Inheritance is advantageous when writing style sheets. For example, if you want all text elements to be blue, write a separate style rule for every element in the document and set the color to

"blue". A better way would be to write a single style rule that applies the color property to the body element and let all the elements contained in the body inherit that style.

Note that, any property applied to a specific element overrides the inherited values for that property.

Conflicting Styles: The Cascade

CSS allows you to apply several style sheets to the same document, which means there are bound to be conflicts. For example, what should the browser do if a document's imported style sheet says that h1 elements should be red, but its embedded style sheet has a rule that makes h1s purple? The two style rules with h1 selectors have equal weight, right?

The cascade refers to what happens when several sources of style information vie for control of the elements on a page: style information is passed down ("cascades" down) until it is overridden by a style rule with more weight. Weight is considered based on the priority of the style rule source, the specificity of the selector, and rule order.

Priority

If you don't apply any style information to a web page, it renders according to the browser's internal style sheet. We've been calling this the default rendering; the W3C calls it the user agent style sheet. Individual users can apply their own styles as well (the user style sheet, also called the reader style sheet), which override the default styles in their browser. However, if the author of the web page has attached a style sheet (the author style sheet), that overrides both the user and the user agent styles. The sidebar "Style Rule Hierarchy" provides an overview of the cascading order from highest to lowest priority.

The only exception is if the user has identified a style as "important," in which case that style will override all competing styles (see the "Assigning Importance" sidebar). This permits users to keep settings accommodating a disability such as extra large type for sight impairment.

Specificity

It is possible for conflicts to arise in which an element is getting style instructions from more than one rule. For example, there may be a rule that applies to paragraphs and another rule for a paragraph that has the ID "intro." Which rule should the intro paragraph use? When two rules in a style sheet conflict, the type of selector is used to determine the winner. The more specific the selector, the more weight it is given to override conflicting declarations. In our example, the

selector that includes the ID name (#intro) is more specific than a general element selector (like p), so that rule would apply to the "intro" paragraph, overriding the rules set for all paragraphs.

It's a little soon to be discussing specificity because we've looked at only two types of selectors. For now, put the term specificity and the concept that some selectors have more "weight," and therefore override others, on your radar.

Assigning Importance

If you want a rule not to be overridden by a subsequent conflicting rule, include the !important indicator just after the property value and before the semicolon for that rule.

For example, to guarantee paragraph text will be blue, use the following rule:

```
p {color: blue !important;}.
```

Even if the browser encounters an inline style later in the document (which should override a document-wide style sheet), like this one:

```
<p style="color: red">
```

that paragraph will still be blue because the rule with the !important indicator cannot be overridden by other styles in the author's style sheet. The only way an !important rule may be overridden is by a conflicting rule in a reader (user) style sheet that has also been marked !important.

This is to ensure that special reader requirements, such as large type or high-contrast text for the visually impaired, are never overridden. Based on the previous examples, if the reader's style sheet includes this rule p {color: black;} the text would still be blue because all author styles (even those not marked !important) take precedence over the reader's styles.

However, if the conflicting reader's style is marked !important, like this

```
p {color: black !important;}
```

the paragraphs will be black and cannot be overridden by any author-provided style.

Beware that the ***!important*** indicator is not a get-out-of-jail free card. Best practices dictate that it should be used sparingly, if at all, and certainly never just to get yourself out of a sticky situation with inheritance and the cascade.

Rule order

After all the style sheet sources have been sorted by priority, and after all the linked and imported style sheets have been shuffled into place, there are likely to be conflicts in rules with equal weights. When that is the case, the order in which the rules appear is important. The

cascade follows a "last one wins" rule. Whichever rule appears last has the last word. Within a style sheet, if there are conflicts within style rules of identical weight, whichever one comes last in the list "wins." Take these three rules,
for example:

```
<style>
p { color: red; }
p { color: blue; }
p { color: green; }
</style>
```

In this scenario, paragraph text will be green because the last rule in the style sheet—that is, the one closest to the content in the document—overrides the earlier ones. Procedurally, the paragraph is assigned a color, then assigned a new one, and finally a third one (green) that gets used. The same thing happens when conflicting styles occur within a single declaration stack:

```
<style>
p { color: red;
color: blue;
color: green; }
</style>
```

The resulting color will be green because the last declaration overrides the previous two. It is easy to accidentally override previous declarations within a rule when you get into compound properties, so this is an important behaviour to keep in mind.

Practical Exercise 2

1. Explain what happens when style sheet rules from different sources come into play?

   _____

2. Embed the CSS code in step (iii) into HTML code in step (i) to apply the defined style to the HTML content.

   _____

3. Explain the steps to run the combined HTML and CSS code as a single program.

   _____

[Attach additional sheet]

Unit 3: CSS text, font, colors

From the styling codes presented in previous paragraph in Unit 2, example of CSS text, font and color is named in the code. To present it in the HTML code, an example is illustrated as follows:

```
font-family: sans-serif;
}
h1 {
text-align: center;
font-family: serif;
font-weight: normal;
text-transform: uppercase;
border-bottom: 1px solid #57b1dc;
margin-top: 30px;
}
h2 {
color: #d1633c;
font-size: 1em;
}
```

Validating documents

To validate a document is to check your markup to make sure that it conforms with all the rules of whatever version of HTML in use. Documents that are error-free are said to be valid. It is strongly recommended documents are validated, especially for professional sites. Valid documents are more consistent on a variety of browsers, they display more quickly, and they are more accessible.

Always use a **validator e.g. html5.validator.nu**. This software checks the source against the HTML version being specified by the developer. Some of the things validator checks for are as follows:

- The inclusion of a DOCTYPE declaration. It defines the version of HTML to validate the source code.
- An indication of the character encoding for the document.
- The inclusion of required rules and attributes.
- Non-standard elements.
- Mismatched tags.
- Nesting errors (incorrectly putting elements inside other elements).
- Typos and other minor errors

Files are either uploaded or a link of the Web page that is online is submitted to the validator for checking and generation of report.

Practical Exercise 3

1. Write a complete program to test **CSS text, font, colors.**

   _____

2. Subject your codes to validation and outline its procedures

   _____

   [Attach additional sheet of paper]

Unit 4: CSS borders, margin and padding, Using CSS in HTML codes

CSS borders, margins and padding can be performed in the HTML codes. Examples of such specification is given as follows

To add a 100-pixel left margin to paragraph (p) elements a declaration:

margin-left: 100px;

To add a 100-pixel left margin to the h2 headings as well. And an orange, 1-pixel border to the bottom of the h1 element a declaration:

border-bottom: 1px solid orange;

To move the image to the right margin, and allow text to flow around it with the float property using the CSS rules:

```
img {
float: right;
margin: 0 12px;
}
```

Practical Exercise 4

1. Based on the HTML and styling codes presented in step (i – iii), put the codes together as specified and run the program.

Print the output screen and attached here

2. Adjust other font styles, and colors to present your Web page to define heading in terms of colour, font styles and font size for the Web page elements. (Hints: colour: green, font family: sans-serif; font size: large)
3. Based on the appearance of the Web page, present your line-by-line comments on the code snippets in (2) above.
4. Run the following code snippets

a) Illustrate the concept of inheritance as would be shown on the output.

```
STYLE DOCUMENT (external.css):
…
h1 { color: red }
…
HTML DOCUMENT:
<!DOCTYPE html>
<html>
<head>
<title>…</title>
<style>
@import url(external.css); /* set to red first */
h1 { color: purple;} /* overridden by purple */

</style>
</head>
<body>
<h1 style="color: blue">Heading</h1> /* blue comes last and wins */
…
</body>
</html>
```

b) Based on the experiment on (a), determine whether the narrative about the output here is either True or False?

Module 4
JavaScript

Unit 1:Basics of JavaScript

JavaScript is a lightweight scripting programming language that adds interactivity and custom behaviours to Web pages. It is a client-side scripting language. JavaScript is presented as a way to add interactivity to a page. Whereas the "structural" layer of a page is our HTML markup, and the "presentational" layer of a page is made up of CSS, the third "behavioural" layer is made up of our JavaScript. JavaScript allows highly responsive interfaces that improve the user experience and provide dynamic functionality, without waiting for the server to load up a new page. Example of such functionality include, among others:

i) Suggestion of the complete term a user might be entering in a search box as he types. This is implemented in action on www.google.com

ii) Request for content and information from the server and injecting it into the current document as needed, without reloading the entire page

iii) Show and hide content based on a user clicking a link or heading, to create a "collapsible" content area.

iv)Test for browsers' individual features and capabilities. For example, one can test for the presence of "touch events," indicating that the user is interacting with the page through a mobile device's browser, and add more touch-friendly styles and interaction methods.

v) Fill in gaps where a browser's built-in functionality falls short or add some of the features found in newer browsers to older browsers. These kinds of scripts are usually called shims or polyfills.

vi) Load an image or content in a custom-styled "lightbox"—isolated on the page with CSS—after a user clicks a thumbnail version of the image.

Creating Java Scripts

To embed a script on a Web page, add the code as the content of a script element as shown:

```
<script>
... JavaScript code goes here
</script>
```

NOTE: For documents written in the stricter XHTML syntax, identify the content of the script element as CDATA by wrapping the code in the following wrapper:

```
<script type="text/javascript">
// <![CDATA[
...JavaScript code goes here
```

```
// ]]>
</script>
```

The external method uses the ***src*** attribute to point to a script file (with a *.js* suffix) by its URL. In this case, the script element has no content:

```
<script src="my_script.js"></script>
```

The advantage to external scripts is that you can apply the same script to multiple pages (the same benefit external style sheets offer). The downside, of course, is that each external script requires an additional HTTP request of the server, which slows down performance.

For most scripts, the preferred placement of JavaScript is at the end of the document, before the </body> tag because the browser will be done parsing the document and its DOM structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
</head>
<body>
…contents of page…
<script src="script.js"></script>
</body>
</html>
```

Example:
An example of the complete date and time script by adding JavaScript to a Web page is as follows:

```
<script language="Javascript" type="text/javascript">
now=new Date();
localtime=now.toString();
utctime=now.toGMTString()
document.write("<b>Local time:</b> " + localtime + "<BR>");
document.write("<b>UTC time:</b> " + utctime);
</script>
```

Example:
Hello World example.

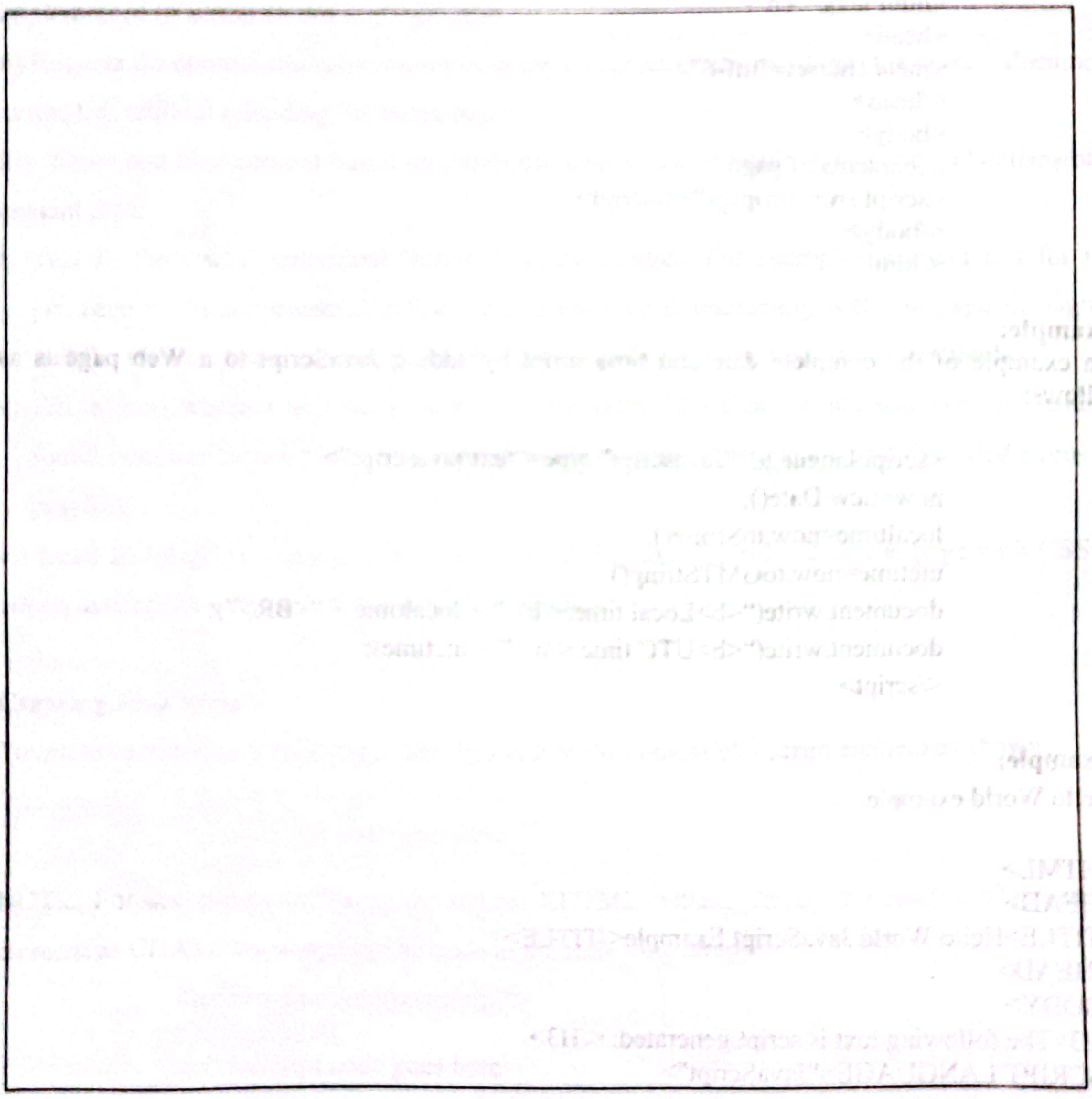
```
<HTML>
<HEAD>
<TITLE>Hello World JavaScript Example</TITLE>
</HEAD>
<BODY>
<H3>The following text is script generated: </H3>
<SCRIPT LANGUAGE="JavaScript">
```

```
<!—
/* Our script only requires
one quick statement! */
document.write("Hello World!") // Prints words to Web document
// end hiding-->
</SCRIPT>
</BODY>
</HTML>
```

The following text is script generated:

Practical Exercise 1

Embed the JavaScript in this example into your HTML codes to show date and time on a Web page. Paste the output of your Web page here.



Unit 2:Using Arithmetic operators in JavaScript

The arithmetic operators available in JavaScript is tabulated as follows:

Operators	Name	Examples
**	exponentiation	2**3
*	multiplication	2*3
/	division	5/4
%	Division remainder	5%4
+	addition	5+3
-	subtraction	5-2
+	concatenation	"John"+"son"

Other operators supported by JavaScript are: Logical, Bit-wise, shift operators, comparison, relational, conditional (ternary), and assignment operators. The numbers used in an arithmetic operation are called operands. The operation is defined by an operator and operands.

Example:

a) let a = 1;
   let b = 2;
   let c = a+ b;

b) let x = 5;
   let z = x ** 2;

Practical Exercise 2

1. Divide 10 by 2, and alert the result.
2. Define the order of arithmetic operators precedence
3. let x = 5;
   let y = 2;
   let z = x / y; and alert the result.

Functions

The basic building block of a script in JavaScript is the function. A function is basically a "mini-program." Functions start by being "passed" by a particular value; they work with that value to make something else happen and then "return" a new value to the body of the program.

In JavaScript, there are two times you need to work with functions. First, you need to Declare the Function. This means that you are defining how the function will work.

The second step is to Call the Function in the body of your script. Generally, your script will be just a series of function calls.

Declaring the Function

A good rule, although it's not necessary, is to declare your functions in the head of your document. The function declaration needs to appear between <SCRIPT> tags, but you can have more than one set of <SCRIPT> tags in an HTML document. Function declaration:

```
<SCRIPT>
<!—
function function_name(value_name)
 {
...function code...
return (new_value)
 }
// end hiding -->
</SCRIPT>
```

Calling a Function

We call the function from the body of the script, which is generally in the body of the document. It does not matter where we declare functions (although, as mentioned, it's best to declare them between the <HEAD> tags), but it is best to put the function calls of the script close to the parts of the document where they are needed. A function call is basically formatted like the following (and always appears between <SCRIPT> tags):

```
function_name(value);
```

In this function call, the function_name should be the same function name that was used in the function declaration, while the value can be anything that is to be passed to the function.

Example

The following example defines the displayHello function:

```
function displayHello() {
document.write( "Hello" );
```

}

The function operator requires the following: function displayHello() -- Function name followed by list of arguments

{ Open Brace document.write("Hello");

-- Sequence of JavaScript statements forming the

}

Example

This example is a JavaScript function to add two numbers:

```
function add() {
 document.write( 5+5 );
 }
add();
```

Another example of functions to perform various arithmetic operations:

```
function minus() {
 document.write(" " + (6-4) );
}
function times() {
 document.write("" + 6*4 );
}
add();
minus();
times()
```

Unit 3: Control Statements in JavaScript

Statements

The basic units of JavaScript program are statements. Statement performs a single action.

Examples of statements are:

hours = now.getHours();

mins = now.getMinutes

secs = now.getSeconds();

Examples of control statements are:

Loop statements

i) **For** statement

```
for (initialize the variable; test the condition; alter the value; ) {
// do something
}
```

Example

```
for (let i = 0; i < cars.length; i++) {
 text += cars[i] + "<br>";
}
```

ii) **While** statement

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

Example

```
while (i < 10) {
 text += "The number is " + i;
 i++;
}
```

iii) **Do** statement

Selection statement

if Statement

This is conditional statement used to specify a block of JavaScript code to be executed if a condition is true. The syntax is as follows:

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

Example

```
if (total_score < 40) {
 grade = "Fail";
}
```

***else* Statement**

This is a conditional statement used to specify a block of code to be executed if the condition is false.

```
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

```
if (hour < 40) {
```

```
  grade = "Fail";
} else {
  grade = "Pass";
}
```

else if Statement

The else if statement is used to specify a new condition if the first condition is false. the syntax is as follows:

```
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

```
if (total_score < 40) {
  grade = "Fail";
} else if (total_score < 45) {
  grade = "E";
} else if (total_score < 50) {
  grade = "D";
} else if (total_score < 60) {
  grade = "C";
} else if (total_score < 70) {
  grade = "B";
}
else {
  grade = "A";
}
```

Switch statement

The switch statement is used to select one of many code blocks to be executed. The syntax is as follows:

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

Example:

```
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";
}
```

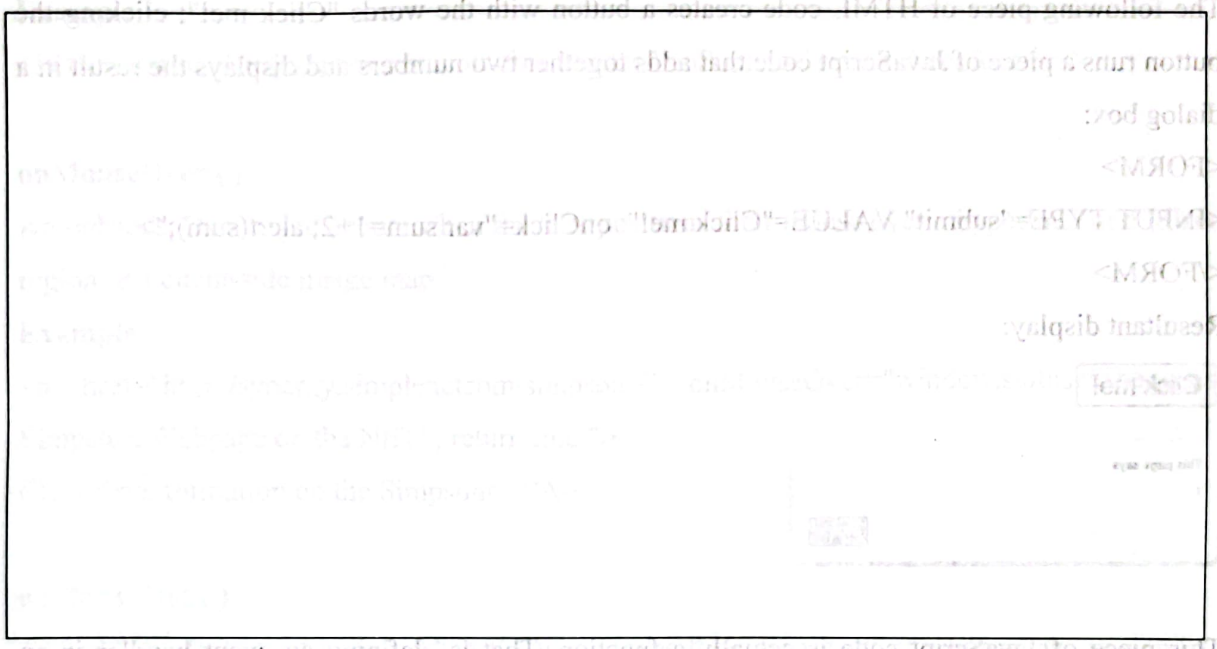
Practical Exercise 3

1) Run the following code snippet to Test JavaScript Random Number Function.

```
<html>
<head>
<title>Math example</title>
</head>
<body>
<h1>Math example</h1>
<p>How random are JavaScript random numbers?
Let generate 5000 and find out how they are valid.</p.>
<script language="JavaScript" type= "text/javascript">
total=0;
for (i=1; i<=5000; i++){
        num=math.random();
        total+= num;
        if (i%1000==0)
                document.write("Generated"+i+"numbers…<br>");
average=total/5000;
average=Math.round(average*1000)/1000;
document.write("<H2>Average of 5000 numbers: "+average+ "</H2>");
```

</script>
</body>
</html>

2) Explain the functions of the control statement as used in the code snippet.
3) Past your output here.



Unit 4:Event handlers in JavaScript

An event handler is a special-purpose function, one that is used only in client-side JavaScript. Event-handler functions are defined unusually; they are defined as fragments of JavaScript within the HTML tags of certain elements on a web page. They are used to handle and verify user input, user actions, and browser actions. Examples of these actions are:

i) Things expected to be done every time a page loads, or closes.
ii) Actions expected when a user clicks a button
iii) Content that should be verified when a user inputs data

JavaScript works with HTML and CSS events. Common events associated with HTML are tabulated thus:

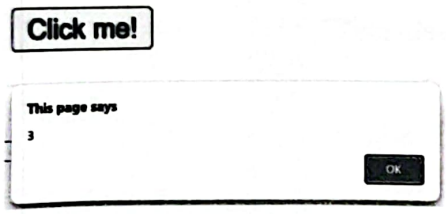
Event	Description
Onchange	An HTML element has been changed
Onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element

onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

The following piece of HTML code creates a button with the words "Click me!"; clicking the button runs a piece of JavaScript code that adds together two numbers and displays the result in a dialog box:

```
<FORM>
<INPUT TYPE="submit" VALUE="Click me!"  onClick="var sum=1+2; alert(sum);">
</FORM>
```

Resultant display:



This piece of JavaScript code is actually a function. That is, defining an event handler in an HTML tag does create a JavaScript function object, just as other function definitions do, and this object can be used as other function objects are. The main difference is that the function will be invoked automatically by the browser in response to appropriate user actions.

The basic format of an event handler is:

<HTML_TAG OTHER_ATTRIBUTES eventHandler="JavaScript Program">

While any JavaScript statements, methods, or functions can appear inside the quotation marks of an event handler, typically, the JavaScript script that makes up the event handler is actually a call to a function defined in the header of the document or a single JavaScript command.

The available event handlers in JavaScript are:

onClick( )

A click event occurs when a button, checkbox, radio button, reset button, or submit button is clicked. This event is regularly used with a button to start script execution.

Example

```
<INPUT TYPE="Button" VALUE="Click Me" onClick = "window.alert('You Clicked me');">
```

onSubmit( )

A submit event occurs when the user submits a form. This event is regularly used with a form and a submit button to start the form validation scripts.

Example

<FORM action="http://www.xnu.com/formtestasp" onSubmit="return checkform();">

onMouseOver ( )

An onMouseOver event occurs when the user positions their mouse over a hyperlink, or a linked region *of* a client-side image map.

Example

<a href="http://synergy.simplenetcom/simpsons/" onMouseOver="window.status='The best Simpsons Webpage on the NET!'; return true;">
Click for information on the Simpsons. </A>

onMouseOut ( )

An onMouseOut event occurs when the user moves their mouse *off of* a hyperlink, or a linked region *of* a client-side image map.

<a href="http://synergy.simplenetcom/simpsons/" onMouseOut="window. status='The best Simpsons Webpage on the NET!'; return true;">
Click for information on the Simpsons.</A>

onFocus ( )

This event occurs when a user tabs into or clicks on a password field, a text field, a textarea, or a FileUpload field in an HTML form. If a user clicks on one of these elements in a form, it is receiving the user's focus.

Example

<INPUT TYPE="TEXT" NAME="Month" onFocus="window.status= ('Please enter the Month as two digits 01 through 12'); return true;">

onLoad()

The load event triggers when the browser finishes loading a document or all the frame pages within a <FRAMESET>.

Example

```
<BODY onLoad="alert('Welcome to our website!');">
```

onUnload( )

The unload event occurs when you move to a new document For example if you use the back button, or click on a link to another page the unload event will occur.

Example

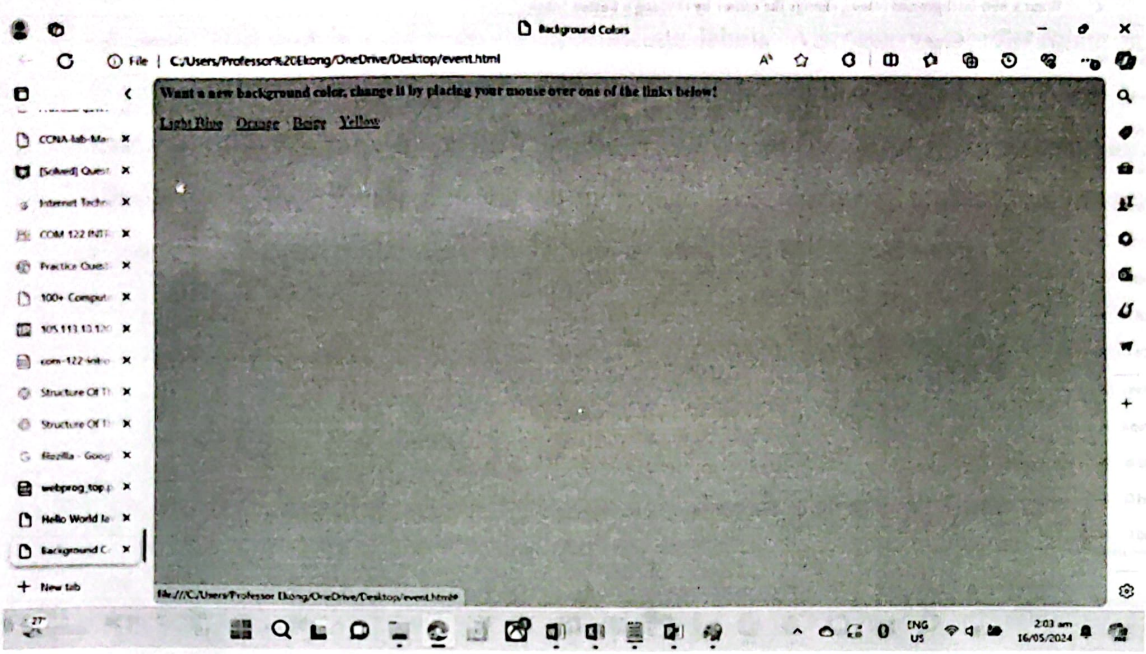
```
<BODY onUnload="alert('Thanks for checking out our site!');">
```

Example

Run the following JavaScript.

```
<HTML>
<HEAD>
<TITLE>Background Colors</TITLE>
<SCRIPT language="JavaScript">
<!--
function newbg(thecolor) {
document.bgColor=thecolor
}
//-->
</SCRIPT>
 </HEAD>
<BODY bgColor = "yellow">
<B>Want a new background color, change it by placing your mouse over one of the links below!</B> <p>
<A HREF="#" onMouseover="newbg( 'lightblue' ) ;">Light Blue</A>   
<A HREF="#". onMouseover="newbg( 'orange') ; ">Orange</A>   
<A HREF="#" onMouseover="newbg('beige') ;">Beige</A>   
<A HREF="#" onMouseover="newbg('yellow') ; ">Yellow</A >
</BODY>
</HTML>
```

The resultant display;



Example

Run the following JavaScript

```
<HTML>
<HEAD>
<TITLE>Background Colors</TITLE>
<SCRIPT language="JavaScript">
<!--
function newbg(thecolor){
document.bgColor=thecolor;
}
</SCRIPT>
</HEAD>
<BODY bgColor="yellow">
<B>Want a new background colour, change the colour by clicking a button below! </B> <P>
<FORM>
<INPUT TYPE= "button" value="Light Blue" onClick="newbg('lightblue');"   
<INPUT TYPE= "button" value="Orange" onClick="newbg('orange') ;">   
<INPUT TYPE= "button" value="Beige" onClick="newbg('beige');">   
<INPUT TYPE= "button" value="Yellow" onClick="newbg('yellow') ;">
</FORM> </BODY>
</HTML>
```

Resultant display:

Practical Exercise 4

1. Run the following code and describe the functions of its JavaScript Event handlers.

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

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can change HTML attribute values.</p>

<p>In this case JavaScript changes the value of the src (source) attribute of an image.</p>

<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Turn on the light</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Turn off the light</button>

</body>
</html>
```

2. Create a form, named calcForm, with two input text fields for length and width, with appropriate labels. In the same form, create two output text fields, named areaBox and perimeterBox and labeled with the appropriate labels. Also create the Calculate button. With an onClick event handler which will call a function named calculate( ). This function will calculate the area and perimeter of the rectangle using the inputted numbers. Finally, the results for the area and perimeter should be displayed in the output text fields. Copy and paste the output for 1 and 2 here or attach an additional sheet.

Module 5

Extensible Markup Language (XML)

Unit 1: XML basics

XML stands for **Extensible Markup Language**. It is a text-based markup language derived from Standard Generalized Markup Language (SGML). It is used as a standard format for data exchange between applications over the Internet.

XML creates markup or tags for describing data of any type of information. The markup describes mathematical formulae (like in Latex), software configuration instructions, chemical molecular structures, music, news, recipes and financial reports. So, XML describes data in a way that both humans and computers can understand.

XML identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data.

Characteristics

a. **XML is extensible** – XML allows you to create your own self-descriptive tags, or language, that suits your application.

b. **XML carries the data, does not present it** – XML allows you to store the data irrespective of how it will be presented.

c. **XML is a public standard** – XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

XML Usage

- XML simplifies the creation of HTML documents for large web sites.
- XML is used to exchange the information between organizations and systems.
- XML is used for offloading and reloading of databases.
- XML is used to store and arrange data, which can customize data handling needs.
- XML can be merged with style sheets to create any desired output.
- XML can represent any type of data.

Example

A simple XML document that describes a university student cumulative grade point average (CGPA) is shown in Figure 1.

```
1     <?xml version = "1.0"?>
2
3     <! - -Fig. 1: student.xml -->
4     <! - -University Student described with XML -->
5     <student>
6           <firstName>Inemesit</firstName>
7           <lastName>Aniekan</lastName>
8           <deptName>Computer Science</deptName>
9           <cGPA>3.75</cGPA>
10    </student>
```

Figure 1: XML that describes a student's CGPA information

The XML documents contain text that represents data such as Inemesit (line 6) and elements that specify the document's structure such as firstName (line 6). XML documents delimit elements with start tag < > and end tags </ > (e.g. <student> and </student>). An element's start and end tags enclose text that represents a piece of data (e.g the student's firstName – Inemesit in line 6 which is enclosed by <firstName> start tag and </firstName> end tag). Every XML document must have exactly one root element that contains all the other elements. The root element in Figure 1 is student.

Practical Exercise 1

Create a simple XML document representing a book with attributes like title, author, and publication year.

Unit 2: Formatting and manipulating XML documents

The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use. XML documents must contain one **root** element that is the **parent** of all other elements. The rules are:

- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.
- The Parameter names and values are case-sensitive.
- The names are always in lower case.
- The order of placing the parameters is important. The correct order is: *version, encoding and standalone.*
- Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. **</?xml>**

XML Syntax declaration

<?xml>

version = "version_number"

encoding = "encoding_declaration"

standalone = "standalone_status"

?>

<?xml version = "1.0">

XML declaration with all parameters defined –

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

XML declaration with all parameters defined in single quotes –

<?xml version = '1.0' encoding = 'iso-8859-1' standalone = 'no' ?>

Practical Exercise 2

Provide a practical example of how to use XML comments effectively to document sections of an XML document.

Unit 3:Viewing and modifying XML documents

This unit describes the various methods to view an XML document. An XML document can be viewed using a simple text editor or any browser. Most of the major browsers supports XML. XML files can be opened in the browser by just double-clicking the XML document (if it is a local file) or by typing the URL path in the address bar (if the file is located on the server), in the same way as we open other files in the browser. XML files are saved with a **".xml"** extension.

Let us explore various methods by which we can view an XML file. Following example (sample.xml) is used to view all the sections of this chapter.

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

Text Editors

Any simple text editor such as Notepad, TextPad, or TextEdit can be used to create or view an XML document as shown in Figure 2.

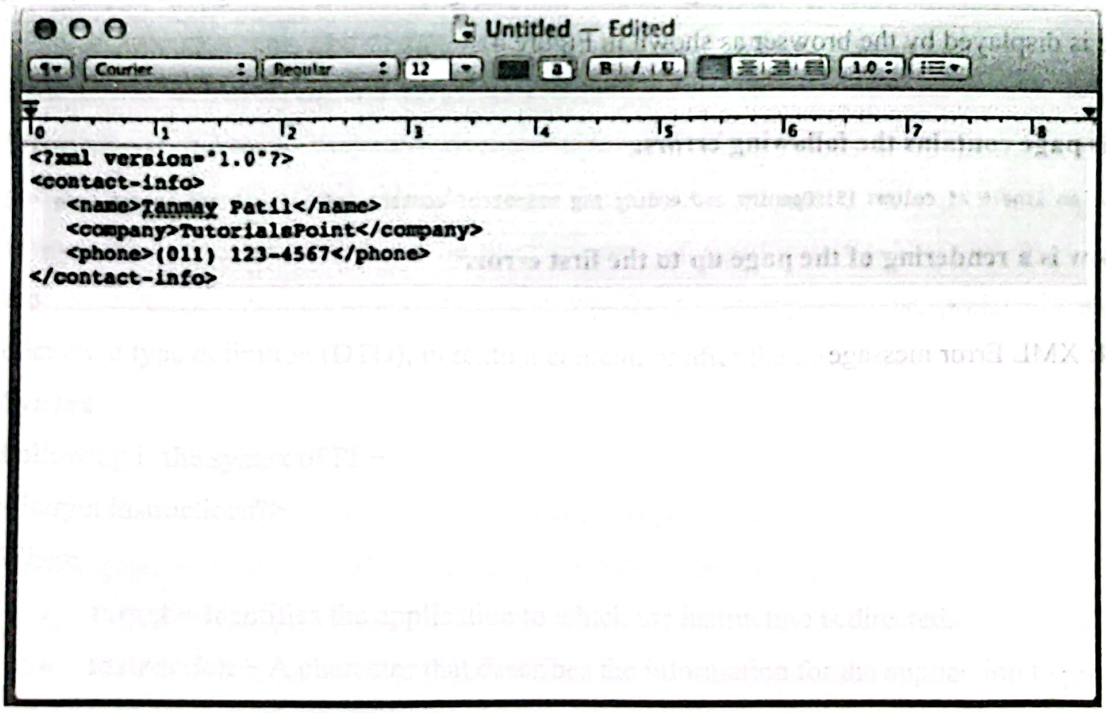


Figure 2: A textpad editor

Chrome Browser

Open the above XML code in Chrome browser. The code gets displayed as shown in Figure 3.

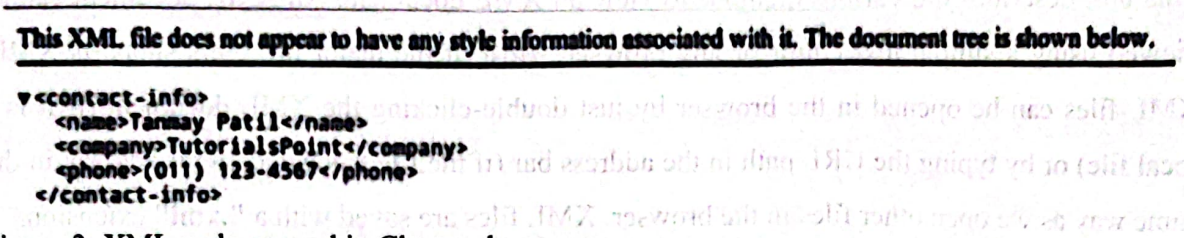


Figure 3: XML code opened in Chrome browser

Errors in XML Document

If the XML code has some tags missing, then a message is displayed in the browser. For example if we try to open the following XML file in Chrome.

```
<?xml version = "1.0"?>
<contact_info>
  <name>Kingsley Joseph</name>
  <company>Kings Systems</company>
  <phone>(234) 805-112-3359</phone>
</contact-info>
```

In the above code, the start and end tags are not matching (contact_info tag), hence an error message is displayed by the browser as shown in Figure 4

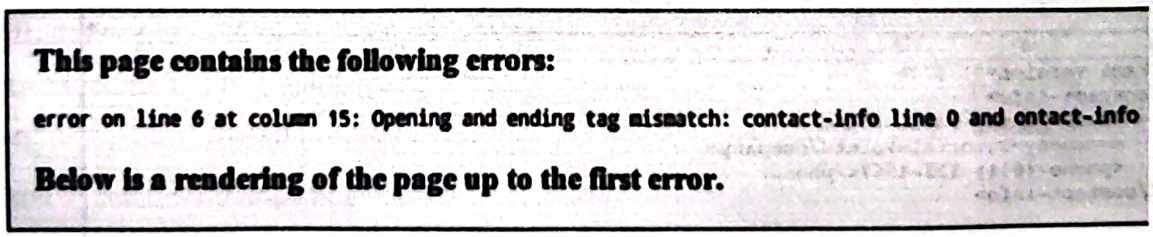
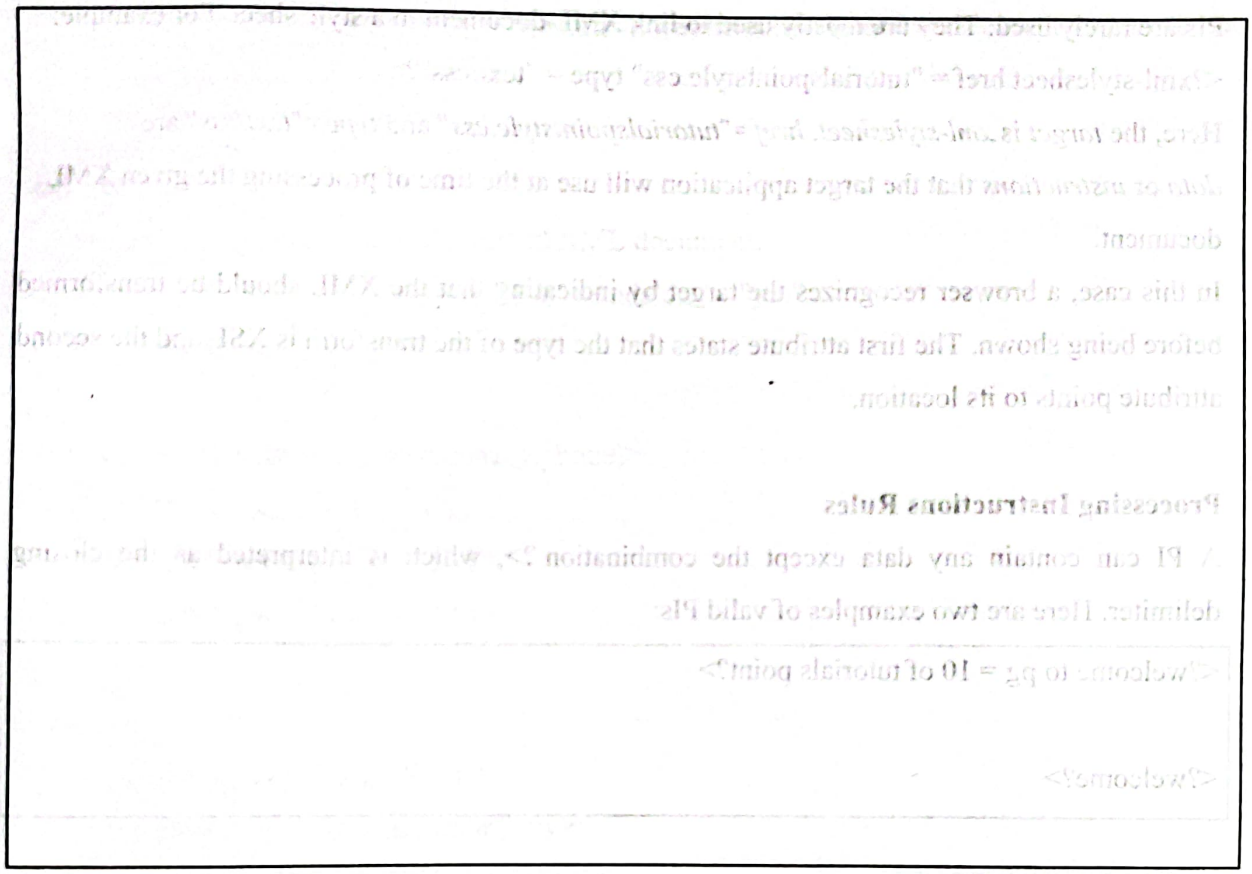


Figure 4: XML Error message

Practical Exercise 3

Add a new element <price> with a value to an existing XML document representing a product catalog.



Unit 4: Processing and validating XML documents

Processing instructions (PIs) allow documents to contain instructions for applications. PIs are not part of the character data of the document, but MUST be passed through to the application. Processing instructions (PIs) can be used to pass information to applications. PIs can appear anywhere in the document outside the markup. They can appear in the prolog, including the document type definition (DTD), in textual content, or after the document.

Syntax

Following is the syntax of PI –

<?target instructions?>

Where

- **target** – Identifies the application to which the instruction is directed.
- **instruction** – A character that describes the information for the application to process.

A PI starts with a special tag <? and ends with ?>. Processing of the contents ends immediately after the string ?> is encountered.

Example

PIs are rarely used. They are mostly used to link XML document to a style sheet. For example;

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

Here, the *target* is *xml-stylesheet*. *href="tutorialspointstyle.css"* and *type="text/css"* are *data* or *instructions* that the target application will use at the time of processing the given XML document.

In this case, a browser recognizes the target by indicating that the XML should be transformed before being shown. The first attribute states that the type of the transform is XSL and the second attribute points to its location.

Processing Instructions Rules

A PI can contain any data except the combination ?>, which is interpreted as the closing delimiter. Here are two examples of valid PIs:

```
<?welcome to pg = 10 of tutorials point?>

<?welcome?>
```

Validating XML Documents

Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration (DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are:

- Well-formed XML document
- Valid XML document

Well-formed XML Document

An XML document is said to be well-formed if it adheres to the following rules:

- Non DTD XML files must use the predefined character entities for **amp(&), apos(single quote), gt(>), lt(<), quot(double quote).**
- It must follow the ordering of the tag (The inner tag must be closed before closing the outer tag).

- Each of its opening tags must have a closing tag or it must be a self-ending tag (<title>....</title> or <title/>).
- It must have only one attribute in a start tag, which needs to be quoted.
- **amp(&), apos(single quote), gt(>), lt(<), quot(double quote)** entities other than these must be declared.

Example

Following is an example of a well-formed XML document:

```
<?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>Kingsley Joseph</name>
    <company>Kings Systems</company>
    <phone>(234) 805-112-3359</phone>
   </address>
```

The above example is said to be well-formed as:

- It defines the type of document. Here, the document type is **element** type.
- It includes a root element named as **address**.
- Each of the child elements among name, company and phone is enclosed in its self-explanatory tag.
- Order of the tags is maintained.

Example

The following is an XML document for an e-mail.

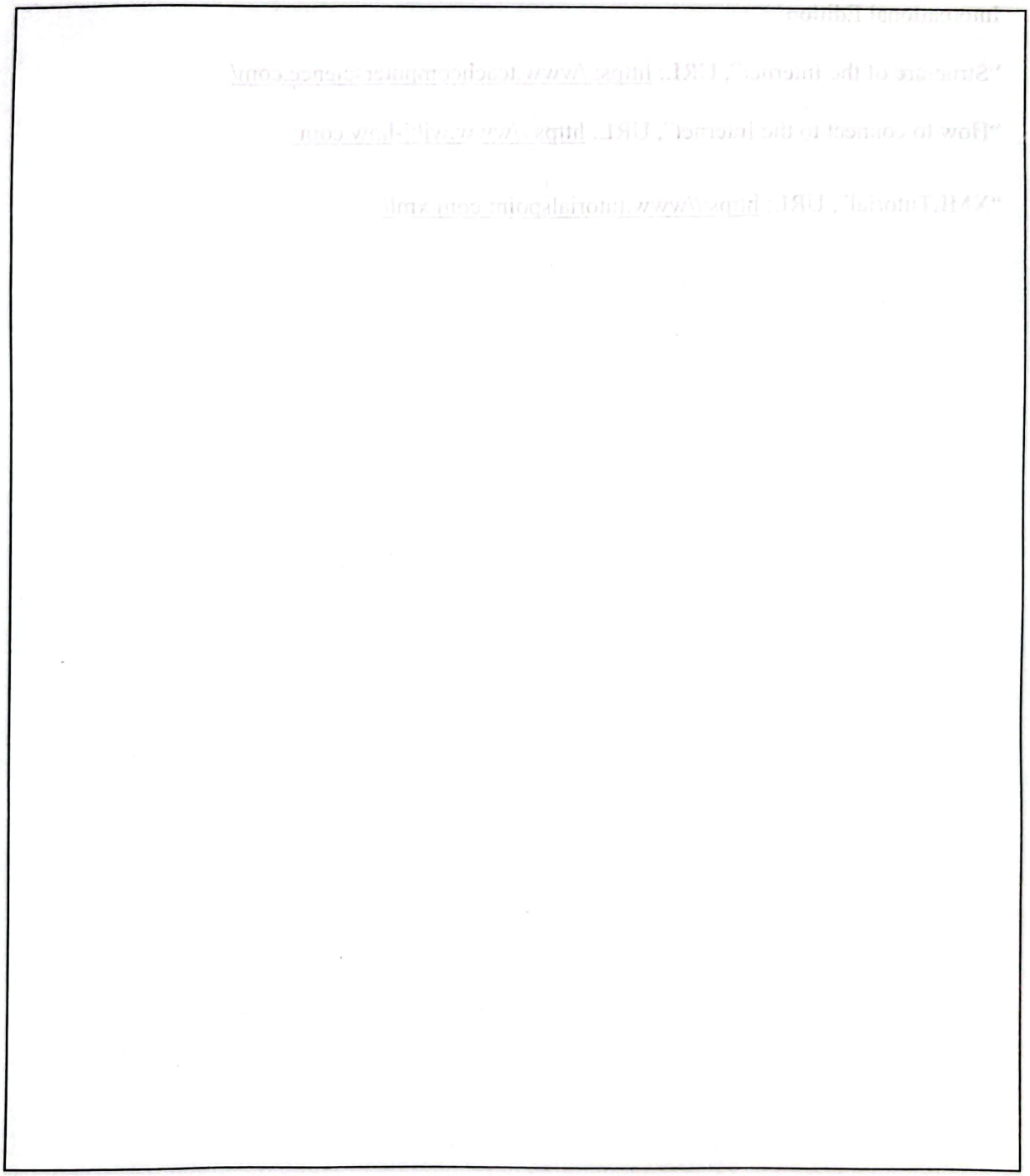
```
<?xml version="1.0" encoding="UTF-8"?>
<emails>
```

```
<email>
 <to>Victor</to>
 <from>Kingsley</from>
 <heading>Hello Sir </heading>
 <body>Hello, how are you sir!</body>
</email>
<email>
 <to>Patience</to>
 <from>Ifiok</from>
 <heading>Birthday wish</heading>
 <body>Happy birthday Ma!</body>
</email>
<email>
 <to>Uyinomen</to>
 <from>Victor</from>
 <heading>Morning walk</heading>
 <body>Please start morning walk to stay fit!</body>
</email>
<email>
 <to>Kingsley</to>
 <from>Patience</from>
 <heading>Health Tips</heading>
 <body>Smoking is injurious to health!</body>
</email>
</emails>
```

Practical Exercise 4

a. Create a DTD to define the structure of an XML document representing a customer profile.

b. Associate the created DTD with an XML document and ensure that it adheres to the defined structure.



References

Ballard, P. and Moncur, M. (2009). Teach Yourself Ajax, JavaScript and PHP. Sams, USA.

Deitel, P. J and Deitel, H. M. (2009) Internet and World wide web: How to program, Pearson International Edition

Terry, F-M. (2009). Web Development and Design Foundations with XHTML USA: Pearson International Edition

"Structure of the Internet", URL: https://www.teachcomputerscience.com/

"How to connect to the Internet", URL: https://www.wiki-how.com/

"XMLTutorial", URL: https://www.tutorialspoint.com/xml/