FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA
NIGER STATE, NIGERIA

CENTRE FOR OPEN DISTANCE AND e-LEARNING (CODeL)

B.TECH. COMPUTER SCIENCE PROGRAMME

COURSE TITLE

INTRODUCTION TO WEB DESIGNING

COURSE CODE

CPT 224

COURSE CODE

CPT 224

COURSE UNIT

2

Course Coordinator

Bashir MOHAMMED (Ph.D.)
Department of Computer Science
Federal University of Technology (FUT) Minna
Minna, Niger State, Nigeria.

Course Development Team

Subject Matter Experts	B. O., ABISOYE (PhD) S. A., ALAGBE O. A., ABISOYE Federal University of Technology, Minna, Nigeria.
Course Coordinator	Bashir MOHAMMED (Ph.D.) Department of Computer Science FUT Minna, Nigeria.
Instructional Designers	Oluwole Caleb FALODE (Ph.D.) Bushrah Temitope OJOYE (Mrs.) Centre for Open Distance & e-Learning, Federal University of Technology, Minna, Nigeria
ODL Experts	Amosa Isiaka GAMBARI (Ph.D.) Nicholas Ehikioya ESEZOBOR
Language Editors	Chinenye Priscilla UZOCHUKWU (Mrs.) Mubarak Jamiu ALABEDE
Centre Director	Abiodun Musa AIBINU (Ph.D.) Centre for Open Distance & e-Learning FUT Minna, Nigeria.

CPT 224 Study Guide

Introduction

CPT 224 Introduction to Web Design is a 2- credit unit course for students studying towards acquiring a bachelor of science in computer science and other related disciplines. The course is divided into 3 module and 7 study units. It will first take a general introduction of the internet and web design, discussing web environment and authoring tool. This course will then go ahead to deal with the different languages used in web page design like HTML, cascading style sheets, server side include, JavaScript. The course went further to explain DHTML, XML, XHTML, WAP and WML. The course also explains the concept of graphics GIF, JPEG, PNG formats, designing graphics with palette, animated GIFs, multimedia and interactivity.

The course guide therefore gives you an overview of what the course. CPT 224 is all about, the textbooks and other materials to be reference, what you expect to know in each unit, and how to work through the course material.

Recommended Study Time

This course is a 3-credit unit course having 15 study units. You are therefore enjoined to spend at least 3 hours in studying the content of each study unit.

What You Are About To Learn In This Course

The overall aim of this course CPT 224 is to introduce you to basic concepts of languages used in web page design and especially its major role in internet. The course defines the languages recognized by web page design. In this course of your studies, you will be put through the definition of common terms in relation to internet and web.

Course Aims

This course aim to introduce student to the basic concept of Internet, Web Server and HTML and appreciate the flexibility of Cascading Style Sheets, Server Side Include and JavaScript endeavours. It will help the reader to understand the level of disparity and relationship between these languages

Course Objectives

It is important to note that each unit has specific objective. Student should study them carefully before proceeding to subsequently unit. Therefore it may be useful to refer to these objectives in the course of your study of the unit to assess your progress. You should always look at the unit objective after completing a unit. In this way, you can be sure that you have done what is required of you by the end of the unit. However, below are overall objective of this course. On completing this course, you should be able to:

i. State why is the need for Internet and Web Page

ii. Define the language recognized by the Web page.

iii. Discuss the history of HTML
iv. Create a good web page using HTML
v. Carry out basic editing on HTML
vi. Identify the HTML structural tags and use the tags correctly
vii. Format text on the web page using HTML,
viii. Create links on the web page,
ix. Discuss the HTML menu tools,
x. Add image and other page elements like font, colour, etc,
xi. Do background editing features in HTML.
xii. A brief History of CSS
xiii. You will learn the basic rules governing CSS
xiv. You will learn about and how to use the different types of CSS
xv. You will learn about Classes and IDs
xvi. Discuss graphics GIF, JPEG and PNG formats,
xvii. Design graphics with palette,
xviii. Add animated GIFs , multimedia and interactivity to your web page.
xix. Definition of JavaScript and relationship between JavaScript and HTML
xx. History of JavaScript and browser incompatibilities
xxi. JavaScript tag, script in head element and body element
xxii. Using an external JavaScript
xxiii. DHTML, XML and uses of XML
xxiv. XML syntax and structure, XML naming rules
xxv. XHTML, different between HTML and XHTML and more,
xxvi. History of WAP,
xxvii. WAP internet model,
xxviii. History of WML,
xxix. How to create link in WML,
xxx. Editing text with WM

Working Through This Course

To complete this course, you are required to study all the units, the recommended text books, and other relevant materials. Each unit contain some Self Assessment Questions s and tutor assignments, and at some point in these courses, you required

to submit the tutor marked assignments. There is also final examination at the end of these courses. Stated below are the component of these courses and what you have to do.

Course Materials

The major components of the course are:

1. Course Guide
2. Study Units
3. Text Books
4. Assignment Files
5. Presentation Schedule

Study Units

There are 7 study unit and 3 modules in this course, They are:

MODULE 1:

UNIT 1: INTRODUCTION TO INTERNET AND WEB SERVER

UNIT 2: CONCEPT OF HTML

MODULE 2:

UNIT 1: FORMATING TEXT, CREATING LINK AND ADDING IMAGES IN HTML

UNIT 2: CASCADING STYLE SHEET AND SERVER SIDE INCLUDES

UNIT 3: GRAPHICS GIF

MODULE 3:

UNIT 1: INTRODUCTION TO JAVASCRIPT, DHTML AND XML

UNIT 2: XHTML, WAP AND WML

Recommended Texts

The following texts and Internet resource links will be of enormous benefit to you in learning this course:

1. Introduction to Web Standards http://dev.opera.com/articles/view/1-introduction-to-the-web-standards-cur/
2. History of the Internet http://dev.opera.com/articles/view/2-the-history-of-the-internet-and-the-w/
3. Responsive Web design http://www.smashingmagazine.com/responsive-web-design-guidelines-tutorials/.
4. Web Design Manual (2010).
5. HTML Utopia by Dan Shafer and Rachel Andrew (2006).
6. The basics of HTML http://dev.opera.com/articles/view/12-the-basics-of-html/.

7. HTML Tutorial http://www.w3schools.com/html/default.asp.
8. HTML Utopia (2006).
9. CSS tutorial http://www.w3schools.com/css/default.asp
10. Creating Cool sites with HTML, XHTML and CSS
11. CSS Basics http://dev.opera.com/articles/view/27-css-basics/
12. CSS- the missing manual 2nd Edition David Swyer Mcfarland
13. Introduction to JavaScript: www.w3schools.com
14. JavaScript Tutorial www.howtocreate.co.uk
15. Introduction to XML: http://www.w3schools.com/xml/xml_whatis.asp
16. Introduction to DHTML: http://www.w3schools.com/dhtml/dhtml_intro.asp

Assignment File

The assignment file will be given to you in due course. In this file you will find all the detail of the work you must submit to your tutor for marks for marking. The marks you obtain for these assignments will count toward the final mark for the course. Altogether, there are tutor marked assignments for this course.

Presentation Schedule

The presentation schedule included in this course guide provides you with importance date for completion of each tutor marked assignment. You should therefore endeavour to meet the deadline.

Assignment

There are two aspects to the assessment of this course. First there are tutor marked assignments: and second the written examination. Therefore you are expected to take note of the fact information and problem solving gathered during the course. The tutor marked assignment must be submitted to your tutor for formal assessments. In accordance to the deadline given. The work submitted will count for 40%of your total course mark. At the end of the course you will need to sit for a final written examination. This examination will account for 60%of your total score.

Tutor-Marked Assignment (TMA)

There are TMAs in this course; you need to submit all the TMAs. The best 10 will therefore be counted. When you have completed each assignment, send them to your tutor as soon as possible and make certain that it gets to your tutor on or before the stipulated deadline. If for any reason you cannot complete your assignment on time, contact your tutor before the assignment is due to discuss the possible of extension. Extension will not be granted after the deadline unless on extraordinary cases.

Final Examination And Grading

Final examination for CPT 224 will last for period of 2 hours and have a values 60% of the total course grade. The examination will consist of questions which reflect the Self Assessment Questions and tutor marked assignment s that you have previously encountered. Furthermore, all areas of the course will be examined. It would be better to use the time between finishing the last unit and sitting for the examination, to revise the entire course. You might find it useful to review your TMAs and comment on them before the examination. The final examination covers information from all parts of the course.

Practical Strategies For Working Through This Course.

1. Read the course guide thoroughly
2. Organize a study schedule. Refer to the course overview for more details. Note the time you are expected to spend on each unit and how the assignment relates to the units. Important details, e.g. details of your tutorials and the date of the first day of the semester are available. You need to gather all this information in one place such as a diary, a wall chart calendar or an organizer. Whatever method you choose, you should decide on and write in your own dates for working on each unit.
3. Once you have created your own study schedule, do everything you can to stick to it. The major reason that students fail that they get behind with their course works. If you get into difficulties with your schedule, please let your tutor know before it is too late for help.
4. Turn to unit 1 and read the introduction and the objectives for the unit.
5. Assemble the study materials. Information about what you need for a unit is given in the table of content at the beginning of each unit. You will almost always need both the study unit you are working on and one of the materials recommended for further readings, on your desk at the same time.
6. Work through the unit, the content of the unit itself has been arranged to provide a sequence for you to follow. As you work through the unit, you will be encouraged to read from your set books.
7. Keep in mind that you will learn a lot by doing all your assignments carefully. They have been designed to help you meet the objectives of the course and will help you pas the examination.
8. Review the objectives of each study unit to confirm that you have achieved them. If you are not certain about any of the objectives, review the study material and consult your tutor.
9. When you are confident that you have achieved a unit's objectives, you can start on the next unit. Proceed unit by unit through the course and try to place your study so that you can keep yourself on schedule.

10. When you have submitted an assignment to your tutor for marking, do not wait for its return before starting on the next unit. Keep to your schedule. When the assignment is returned, pay particular attention to your tutor's comments, both on the tutor marked assignment form and also written on the assignment. Consult your tutor as soon as possible if you have and questions or problems.
11. After completing the last unit, review the course and prepare yourself for the final examination. Check that you have achieved the unit objectives (listed at the beginning of each unit) ant the course objectives (listed in this course guide).

Tutors And Tutorials

There are 8 hours of tutorial provide in support of this course. You will be notified of the dates, time and location together with the name and phone number of your tutor as soon as you are allocated a tutorial group. Your tour tutor will mark and comment on your assignments, keep a close watch on your progress and on any difficulties you might encounter and provide assistance to you during the course. You must mail your tutor marked assignment to your tutor well before the due date. At least two working days are required for this purpose. They will be marked by your tutor and returned to you as soon as possible. Don not hesitates to contact your tutor by telephone, e-mail or discussion board if you need help. The following might be circumstances in which you would find help necessary:

i. You don't not understand any part of the study units or the assigned readings;
ii. You have difficulty with the self-test or exercise.
iii. You have questions or problems with an assignment, with your tutor's comments on an assignment or with the grading of an assignment.

You should endeavour to attend the tutorials. This is the only opportunity to have face to face contract with your tutor and ask questions which are answered instantly. You can raise any problem encountered in the course of your study. To gain the maximum benefit from the course tutorials, have some questions handy before attending them. You will learn a lot from participating actively in discussions.

Good luck!

Table of Content

Module 1

Introduction to Internet and Web Server

Unit 1: Introduction to Internet and Web Server

Unit 2: Concept of HTML

Unit 1

Introduction to Internet and Web Server

Content

1.0 Introduction

We are in the information age; there is a large demand for web developers. The study of Web designing is an important part of the core of Computer Science. While previous programs have prepared you for programming languages here we explore the foundations of the internet, how to Plan a web site and finally we look at two languages that support simple web sites – HTML and CSS.

2.0 Learning Outcomes

At the end of this unit, you should be able to:

i. Narrate the history of internet
ii. List the materials needed to connect to internet
iii. Discuss how to make better use of internet
iv. Explain the web server and its usefulness
v. Discuss the web environment and its usefulness
vi. Explain the authoring tools and its usefulness

3.0 Learning Outcome

3.1 Introduction to the Internet and Web Servers

In the present age of information Technology, use of Internet is now a popular means of accessing information on any topic of your interest. It also provides tremendous opportunities to students; researchers and professionals for getting information on matters related to academic and professional topics and lot more. In the present world, most of the people who have computers around themselves use Internet to access information from the World Wide Web, exchange messages & documents and e-services.

The task that can be performed on the Internet

- Email
- Obtain information.
- Entertainment such as games, radio, reviews of movies.
- Discussion Groups (chat rooms)
- Online Shopping
- Services such as online banking

Items/components needed to be connected to the Internet?

- Personal Computer
- Modem
- Phone Line
- Internet Service Provider
- Web Browser

Modems: Modems are devices used to translate information from a digital to a readable format on the screen. New computers come with them pre-installed. The speed with which they operate determines how fast you will receive information.

Internet Service Providers: An Internet Service Provider can be compared to a long distance phone company where a monthly fee is paid, the computer dials up the ISP, and the ISP connects the customer to the network.

Examples ISP:

- America Online (AOL)
- Globalcom
- Multilink
- Mtn Nigeria
- Earthlink
- MSN
- **Connection Options**
- Dial Up
- Cable Modem
- DSL

Self-Assessment Question(s)

1. In your own words explain what the internet is.
2. In your own words explain the history of the internet.
3. What is the goal of site planning ?
4. List as many web browser you know

Self-Assessment Answer

1. The internet is the interconnection of computer systems over a wide area. It provides opportunities to students; researchers and professionals for getting information on matters related to academic and professional topics and lot more. Most of the people who have computers around themselves use the Internet to access information from the World Wide Web, exchange messages & documents and e-service
2. The internet was first concieved by ARPA currently renamed DARPA (Defence Advanced Research Project Agency) In 1960, psychologist and computer scientist Joseph Licklider published a paper entitled Man-Computer Symbiosis, which articulated the idea of networked computers providing advanced information storage and retrieval. In 1962 he formed a group to further computer research, but left the group before any actual work was done on the idea.

The idea for a planned Computer network (to be called ARPANET) was presented in October 1967, and in December 1969 the first four-computer network was up and running. The creation of multiple networking protocols did more harm than good in the long term. However a worker of DARPA named Robert Kahn and Vinton Cerf from Stanford University created a system that made the different networking protocol and forming a new standard for transmitting over the internet and it was called the **Internet Transmission Control Protocol**

3. The goal of site planning is to build a website that is useful for the people who will be using it.
4. The list of web browsers is as follows:
   - Mozilla Firefox
   - Google Chrome
   - Internet Explorer
   - Safari
   - Netscape Navigator.

3.2 World Wide Web

The World Wide Web is a part of the Internet. It is a collection of millions of pages of information. The internet is the interconnectivity of computer systems over a very large area for communication i.e sending and receiving of data in the form of text, images video and sound. The internet was first concieved by ARPA currently renamed DARPA (Defence Advanced Research Project Agency) which recognised the need for an organisation that could research ideas and technology beyond what was currently needed at the time.

In 1960, psychologist and computer scientist Joseph Licklider published a paper entitled Man-Computer Symbiosis, which articulated the idea of networked computers providing advanced information storage and retrieval. In 1962, whilst working for DARPA as the head of the information processing office, he formed a group to further computer research, but left the group before any actual work was done on the idea.

The plan for this computer network (to be called ARPANET) was presented in October 1967, and in December 1969 the first four-computer network was up and running. The core problem in creating a network was how to connect separate physical networks without tying up network resources for constant links. The technique that solved this problem is known as packet switching and it involves data requests being split into small chunks (packets,) which can be processed quickly without blocking communication from other parties - this principle is still used to run the Internet today.

This idea received wider adoption, with several other networks coming up and using the same packet switching technique—for example, X.25 (developed by the International Telecommunication Union) formed the basis of the first UK university network JANET (allowing UK universities to send and receive files and emails) and the American public network CompuServe (a commercial enterprise allowing small

companies and individuals access to time-shared computer resources, and then later Internet access.) These networks, despite having many connections, were more private networks than the Internet of today.

The creation of multiple networking protocols did more harm than good in the long term and became a problem. However, a worker of DARPA named Robert Kahn and Vinton Cerf from Stanford University created a system that made the different networking protocol and forming a new standard for transmitting over the internet and it was called the ***Internet Transmission Control Protocol.***

After much funding and development of the software from DARPA, it was published adopted for public use in the ARPANET computer network and those outside the US were also converted to the new TCP.

3.2.1 The coming of Web Standards

Microsoft and Netscape focused on implementing new features rather than on fixing problems with the features they already supported, and adding proprietary features and creating features that were in direct competition with existing features in the other browser, but implemented in an incompatible way.

Developers at the time were forced to deal with ever increasing levels of confusion when trying to build web sites, sometimes to the extent of building two different but effectively duplicate sites for the two main browsers, and other times just choosing to support only one browser, and blocking others from using their sites. This was a terrible way of working, and the inevitable backlash from developers was not far away.

3.2.2 The Formation of the W3C

In 1994, Tim Berners-Lee founded the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology, with support from CERN, DARPA (as ARPA had been renamed to) and the European Commission. The W3C's vision was to standardize the protocols and technologies used to build the web such that the content would be available to as wide a population of the world as possible.

During the next few years, the W3C published several specifications (called recommendations) including HTML 4.0, the format for PNG images, and Cascading Style Sheets versions 1 and 2.

However, the W3C did not (and still do not) enforce their recommendations. Manufacturers only had to conform to the W3C documents if they wished to label their products as W3C-compliant. In practice, this was not a valuable selling point as almost all users of the web did not know, nor probably care, who the W3C were (this is still the case, to a large extent). Consequently, the browser wars of the nineties continued unabated.

3.2.3 The Web Standards Project

In 1998, the browser market was dominated by Internet Explorer 4 and Netscape Navigator 4. A beta version of Internet Explorer 5 was released, and it implemented a new and proprietary dynamic HTML. This meant that professional web developers needed to know five different ways of writing JavaScript.

As a result, a group of professional web developers and designers banded together. This group called themselves the Web Standards Project (WaSP). The idea was that by calling the W3C documents standards rather than recommendations, they might be able to convince Microsoft and Netscape to support them.

The early method of spreading the call to action was to use a traditional advertising technique called a roadblock, where a company would take out an advert on all broadcast channels at the same time, so no matter how a viewer would flick between channels, they would get exactly the same message. The WaSP published an article simultaneously on various web development focused sites including builder.com, Wired online, and some popular mailing lists.

Another technique the WaSP used was to ridicule the companies involved with the W3C (and other standards bodies) that focused more on creating new, often self-serving, features rather than working to get the basic existing standards supported correctly in their products to start with (this includes some browser companies that shall remain nameless here). This doesn't mean that the WaSP ridiculed the W3C themselves; rather they ridiculed the companies that became W3C members and then misbehaved.

The W3C has a few full time staff, but most of the people who work on the standards are volunteers from member companies, e.g. Microsoft, Opera, Mozilla, Apple, Google, IBM, Adobe, to name a few of the bigger ones.

This all sounds a bit negative, but the WaSP didn’t just sit their criticising people—they also helped. Seven members formed the CSS Samurai, who identified the top ten problems with the CSS support in Opera and other browsers (Opera fixed their problems, others did not).

3.2.4 The Rise of Web Standards

In 2000, Microsoft released Internet Explorer 5 Macintosh Edition. This was a very important milestone, it being the default browser installed with the Mac OS at the time, and having a reasonable level of support for the W3C recommendations too. Along with Opera's decent level of support for CSS and HTML, it contributed to a general positive movement, where web developers and designers felt comfortable designing sites using web standards for the first time.

The WaSP persuaded Netscape to postpone the release of the 5.0 version of Netscape Navigator until it was much more compliant (this work formed the basis of what is now Firefox, a very popular browser). The WaSP also created a Dreamweaver

Task Force to encourage Macromedia to change their popular web authoring tool to encourage and support the creation of compliant sites.

The popular web development site A List Apart was redesigned early in 2001 and in an article describing how and why, stated:

In six months, a year, or two years at most, all sites will be designed with these standards. We can watch our skills grow obsolete, or start learning standards-based techniques now.

That was a little optimistic—not all sites, even in 2008, are built with web standards. But many people listened. Older browsers decreased in market share, and two more very high profile sites redesigned using web standards: Wired magazine in 2002 and ESPN in 2003 became field leaders in supporting web standards and new techniques.

Also in 2003, Dave Shea launched a site called the CSS Zen Garden. This was to have more impact on web professionals than anything else, by truly illustrating that the entire design can change just by changing the style of the page; the content could remain identical.

Since then in the professional web development community web standards have become de rigeur. And in this series, we will give you an excellent grounding in these techniques so that you can develop websites just as clean, semantic, accessible and standards-compliant as the big companies'.

3.2.5 Web Standards and Model

The basic building blocks of the World Wide Web are the three main web standards - HTML (or XHTML), CSS and JavaScript. Hypertext Markup Language is actually a pretty good name as far as communicating it's purpose. HTML is what's used to divide up a document, specify its contents and structure, and define the meaning of each part (it's what contains all the text etc that you see on web sites.) It uses elements to identify the different components of a page.

Cascading Style Sheets give you complete control over how an element is displayed. It's easy, using style declarations, to change all paragraphs to be double-spaced (line-height: 2em;), or to make all second-level headings green (colour: green;). There are a ton of advantages to separating the structure from the formatting. To demonstrate the power of HTML and CSS used together, Figure 3.1.1 shows some plain HTML on the left, with no formatting added to it at all, while on the right you can exactly the same HTML with some CSS styles applied to it.

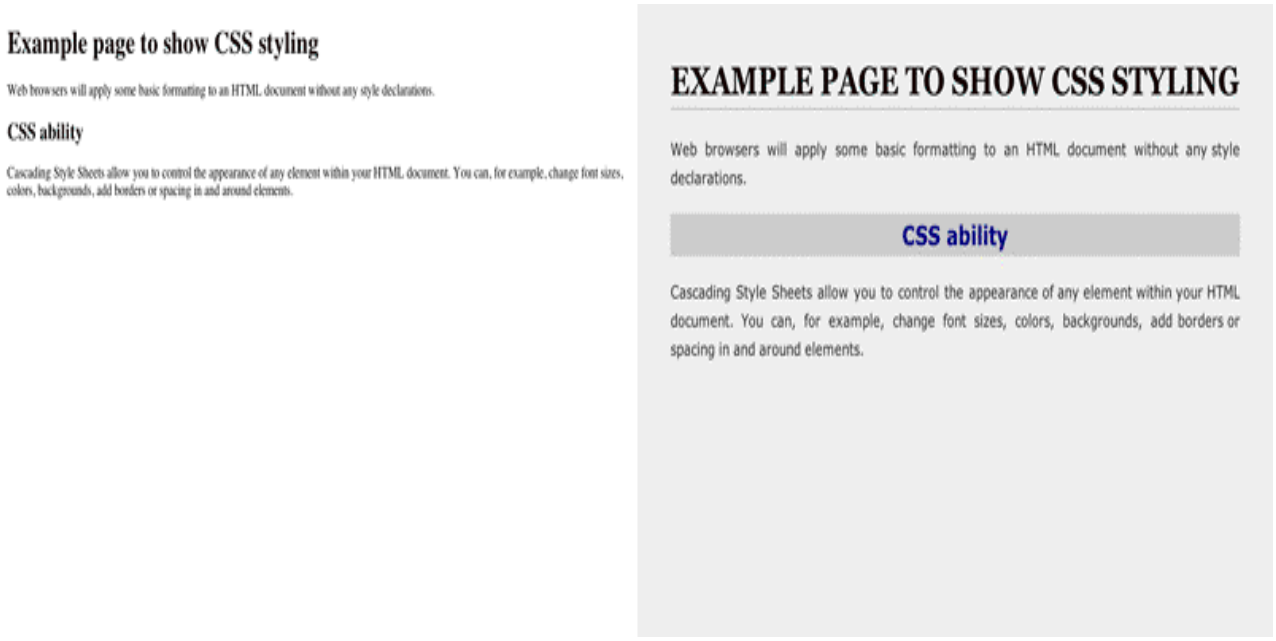


Fig. 3.1.1: Plain HTML on the left, HTML with CSS applied to it on the right.

Finally, JavaScript provides dynamic functions to your web site. You can write small programs in JavaScript that will run on the client computer, requiring no special software to be installed on the server. JavaScript allows you to add some basic functionality and interactivity to your web site, but it has its limitations, which brings us to server-side programming languages, and dynamic web pages.

3.3 Dynamic Web Pages

Sometimes, when browsing the Internet, you'll come across web pages that don't have an .html extension—they might have a .php, .asp, .aspx, .jsp, or some other strange extension. These are all examples of dynamic web technologies, which can be used to create web pages that have dynamic sections—code that displays different results depending on values fed to it, e.g from a database, form, or other data source. We'll cover these types of web pages in the Static versus Dynamic pages section below.

Formats requiring other applications or plugins

Because web browsers are only equipped to interpret and display certain technologies like web standards, if you've requested a URL that points to either a complex file format, or a web page containing a technology requiring plugins, it will either be downloaded to your computer or opened using the required plugin if the browser has it installed. For example:

1. If you encounter a Word document, Excel file, PDF, compressed file (ZIP, or SIT for example,) complex image file such as a Photoshop PSD, or another complex file that the browser doesn't understand, the browser will usually ask you if you want to download or open the file. Both of these usually have similar results, except

that the latter will cause the file to be downloaded and then opened by an application that does understand it, if one is installed.

2. If you encounter a page containing a Flash movie, MP3 or other music format, MPEG or other video format, the browser will play it using an installed plugin, if one has been installed. If not, you will either be given a link to install the required plugin, or the file will download and look for a desktop application to run it.
   Of course, there are some gray areas—for example SVG (Scalable Vector Graphics) is a web standard that runs natively in some browsers, such as Opera, but not in others, such as Internet Explorer—IE needs a plugin to understand SVG. A number of browsers will come with some plugins pre-installed, so you may not be aware that content is being displayed via a plugin and not natively within the browser.

3.3.1 Static vs. Dynamic Web Sites

So what are static and dynamic web sites, and what is the difference between the two? Similar to a box of chocolates, it's all in the filling:

A static web site is a web site where the content, the HTML and graphics, are always static—it is served up to any visitor the same, unless the person who created the web site decides to manually change the copy of it on the server - this is exactly what we've been looking at throughout most of this article.

On a dynamic web site on the other hand, the content on the server is the same, but instead of just being HTML, it also contains dynamic code, which may display different data depending on information you feed to the web site. Let's look at an example - navigate to www.amazon.com in your web browser, and search for 5 different products. Amazon hasn't sent you 5 different pages; it has sent you the same page 5 times, but with different dynamic information filled in each time. This different information is kept in a database, which pulls up the relevant information when requested, and gives it to the web server to insert in the dynamic page.

Another thing to note is that special software must be installed on the server to create a dynamic web site. Whereas normal static HTML files are saved with a file extension of .html, these files contain special dynamic code in addition to HTML, and are saved with special file extensions to tell the web server that they need extra processing before they are sent to the client (such as having the data inserted from the database) - PHP files for example usually have a .php file extension.

There are many dynamic languages to choose from - I've already mentioned PHP, and other examples include Python, Ruby on Rails, ASP.NET and Coldfusion. In the end, all of these languages have pretty much the same capabilities, like talking to databases, validating information entered into forms, etc., but they do things slightly differently, and have some advantages and disadvantages. It all boils down to what suits you best.

Web browsers: A browser is a software program that lets you read and view websites.

Examples:

- Internet Explorer
- Netscape Navigator
- Mozilla Firefox
- Safari
- Google Chrome

3.4 Web Site Planning

Traditionally, the planning stage of a web site (or any project) can be a little stressful. Everyone has an opinion about how a web site should be built, and often their opinions will conflict with one another. Your number one goal on any web site should be to build something that's useful for the people who will be using it. It really doesn't matter what your boss says, what that guy down the hall with a doctorate in software engineering says, or even what your personal preferences are; at the end of the day, if you're building a web site for a particular group of people, their opinion is the only one that matters.

This article is going to look at the early stages of planning out a web site, and a discipline that is commonly referred to as Information architecture, or IA. This involves thinking about who your target audience will be, what information and services they need from a web site, and how you should structure it to provide that for them. You'll look at the entire body of information that needs to go on the site and think about how to break that down into chunks, and how those chunks should relate to one another

You need to plan out the site you're building

You'll come upon the odd web project that you can just dive right into without any up front thought, but these are, by far, the exception and not the norm. We're going to take a look at a fictional band called "The Dung Beatles" and try to help them work through the early stages of planning out their web site. We'll talk with the band and find out what goals they have, and what they would like to see on their web site. Then we'll dive in and start working on a structure for the band's information.

Introducing "The Dung Beatles"

The Dung Beatles (TDB) has a problem. They are the hottest Beatles tribute band in Moose Jaw, Saskatchewan, but they need to raise their profile for an upcoming Northern tour this summer. They've got venues scheduled throughout Canada and the United States, but they're virtually unknown outside of their hometown. If only there was some way, using technology, to reach a large number of Beatles fans for relatively little money.

Lucky for TDB, we've got this thing called the World Wide Web, and they quickly decide that building a web site is the answer they've been searching for. TDB needs a place to promote their tour dates, build a fan base in other cities and raise awareness

of the band. You're going to work through their ideas with them and see if you can chart out a plan for their web site.

You schedule a meeting with your new clients to hash out the details of what they're looking for and to decide on due dates and costs. You open the conversation by suggesting that you talk about the goals and objectives of the web site in order to get an idea of what they want. What does the band hope to achieve with their online presence?

TDB starts talking about their upcoming tour, and how they want to get the word out to Beatles fans in all of their scheduled stops. It's February now, and they're scheduled to kick off their tour in five months time.

Hang on a second! A web site alone won't build it's own traffic and publicise itself. You extract from the conversation thus far that the main goal for the site is to provide a home for TDB fans online; a place where they can keep up to date on the latest news, tour dates and venues. Through the fans (word of mouth), and some other advertising venues, new people will be driven to the web site where they can download sample tracks, check out pictures of the band (in full costume) and find out where/when they can check them out live.

Raul McCoffee, the front man of the group, points out that it would be nice to be able to raise a little extra money for the tour through the sale of some CDs and band merchandise. You gather the band around and draw out a quick sketch of what a visitor might want when they visit the web site. This is just a really rough brainstorm of ideas; it's got very little structure at this point.

There are two general groups of people who will visit the site—people who know TDB already and like them (fans), and people who are unsure. You've got to cater to both those groups in different ways; potential fans need to be "sold" on the group, whereas current fans want to "feed their addiction" (so to speak). What sort of information is each of these groups going to be looking for? Figure 3.1.2 gives an indication of this—this is a typical sketch of the type that you'll want to make at this point in future web site projects. From this, you'll work out what pages the web site needs, and how they should link to one another.

Fig. 3.1.2: Chart of Internet users

You settle on a budget, and agree to launch the web site in one month. You promise to get back to the band in a couple of days with some plans outlining the direction you're going in.

Drawing a Site Map

A lot of people will throw together a site map at this stage—this looks like an organisational chart. This is usually a basic graphic sketch showing simply the names of each page on the site and how they link into the overall structure of the web site. You might want to put in a little more detail and talk about the purpose and content of each page. For example, a page may be labelled "Home", but what is the home page? or is it a more dynamic page containing news items and enticing images? Take a few minutes to think about what pages the above sketch might turn into, and what might be contained on each page. Have a go at drawing your own site map before moving on to the next section.

Now let's get started with the basics: one of those org charts that I mentioned above. The Figure 3.1.3 below shows my attempt at taking the brainstorm and turning it into a site org chart:

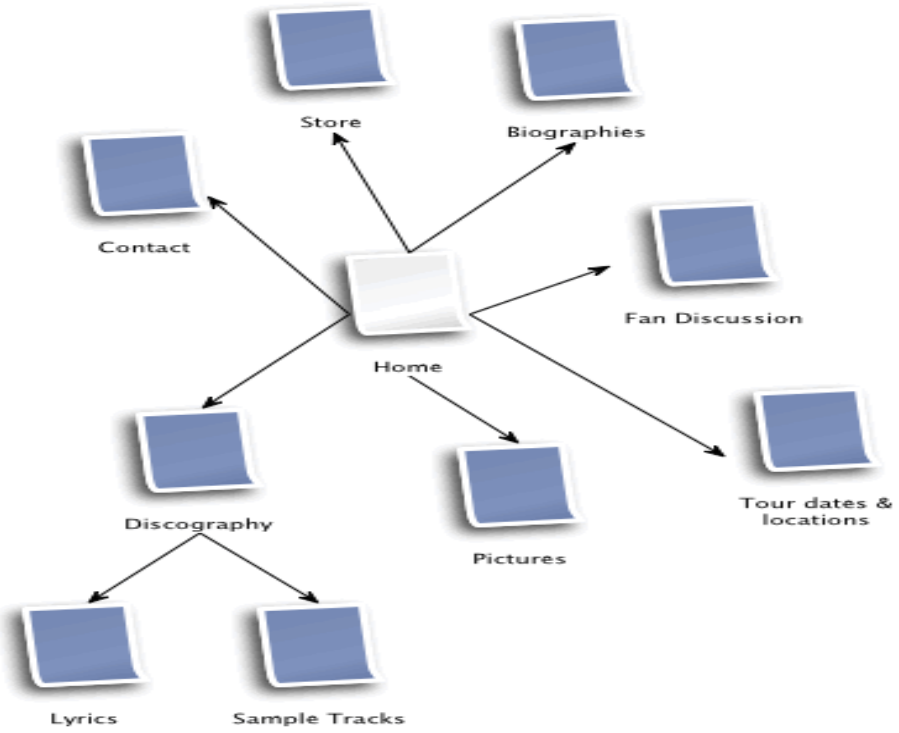


Fig. 3.1.3: Site Org Chart

That definitely captures all of the pages we'll need, but there's no real grouping going on here. It's just a big mess of pages now, and at this point I hadn't really given a lot of thought to what things are called. I did one more pass and try to "chunk" the information into slightly larger groupings—The figure 3.1.4 below the illustration:

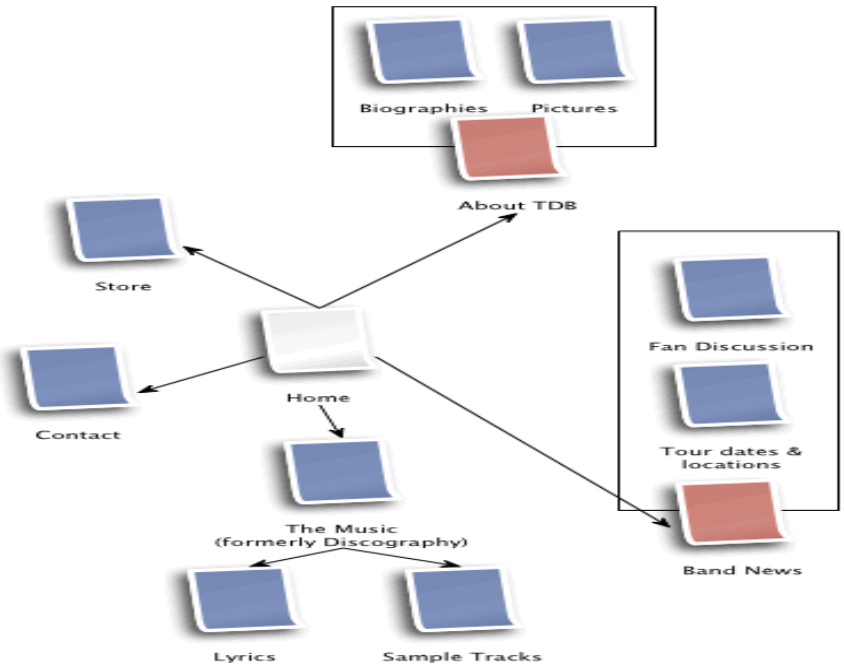


Fig. 3.1.4: Site Structure

I've done a couple of things with the revised site structure. The "Band News" page gives TDB a place to post anything they want to share with their fans. Even after their summer tour is over, and the "Tour dates and locations" page is no longer relevant, they'll be able to post stuff. Adopting a blog format here will let fans comment in context on the various stories, and will help to build an online community around TDB. News and tour events will likely spark the most discussion, so let's group that all together. Additionally, the word "News" is a simpler, more general word that people will be able to recognise faster if they're skimming a page for the information they want.

Our new "About The Dung Beatles" page groups together the band members' biographies as well as their pictures. Going this route gives us a jumping off point for individual band member biographies. Following a similar argument to the one we made above, "About" is a common term used on a lot of web sites. Any time a visitor wants to learn more about a company, a product, a service, or an individual, they usually look for an "About" link.

Finally, the term "Discography" is a bit of a technical term. It's possible that fewer people will understand what that term means than "The Music". Also, it opens up this page to additional content: sources of inspiration, history of a particular song…you get the idea. I think we're ready to roll. After I've talked a bit about naming pages sensibly, we'll move on to add a little more detail about each page.

Naming your pages

Page names can be one of the most crucial decisions you'll make during web site design. Not only is it important for your visitors so that they can find their way around your web site; it is also another thing that dictates how easy your site is to find using a search engine (you'll find various mentions of search engine optimisation throughout the course).

In general, search engines look at the text included in a web page, the URL of that page, and the text of any links to that page when they're deciding "how important" it is. Giving your pages sensible names and sensible URLs will encourage anyone linking to your pages to use sensible descriptions.

Here's an example. Let's assume that you own a car company, and you have a model called "The Speedster". Let's assume you have a web site to promote your automobile, and one of the pages lists available features. Do you call this page "Features", "Available Features", "Features of the Speedster", or "Bells and Whistles"? I would suggest that "Features of the Speedster" is the best option from this list. It's specific to what the page contains, chances are that the title will be displayed high up on the page and will be prominent (good for search engine indexing), and you may even be able to fit it into the URL (something like "www.autocompany.com/speedster/speedster-features/").

Adding some details

You don't have to figure out everything at this point, but you need to at least provide a brief description of what you have in mind for each page. After you've got the site

structure, number each of your pages and provide a brief description for each page, like I've done in Figure 3.1.5 for the home page (you'll get a chance to do this for the other pages in one of the exercises questions at the end of the article.)

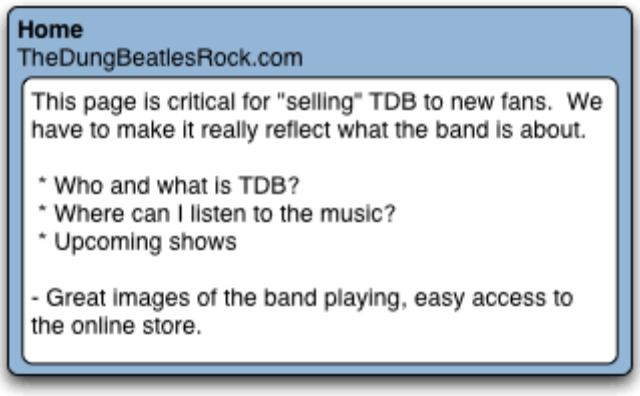


Fig. 3.1.5: TDB Home Page

This is about as involved as you want to get at this point. You don't need to describe page functionality, the technology you'll use to build it, or the design/layout in great detail. Just describe what you have in mind in general terms. Your goal here is to communicate what you're thinking to your client and to force you to think things through.

It's not uncommon at this stage to come to the realisation that you have too many pages, and you'll never be able to find content for them. You can go crazy in creating a hierarchy of pages. For example, if the band members just wanted to publish one paragraph about themselves, it wouldn't be necessary to create separate biography pages for each member. They could all be combined into a single page.

Web Server: A web server is the combination of computer and the program installed on it. Web server interacts with the client through a web browser. It delivers the web pages to the client and to an application by using the web browser and the HTTP protocols respectively. We can also define the web server as the package of large number of programs installed on a computer connected to Internet or intranet for downloading the requested files using File Transfer Protocol, serving e-mail and building and publishing web pages. A web server works on a client server model. A computer connected to the Internet or intranet must have a server program. While talking about Java language then a web server is a server that is used to support the web component like the Servlet and JSP. Note that the web server does not support to EJB (business logic component) component.

A computer connected to the Internet for providing the services to a small company or a departmental store may contain the HTTP server (to access and store the web pages and files), SMTP server (to support mail services), FTP server (for files downloading) and NNTP server (for newsgroup). The computer containing all the above servers is called the web server. Internet service providers and large companies may have all the servers like HTTP server, SMTP server, FTP server and many more

on separate machines. In case of Java, a web server can be defined as the server that only supports to the web component like servlet and jsp. Notice that it does not support to the business component like EJB

Self-Assessment Question(s)

1. In your own words explain what the internet is.
2. In your own words explain the history of the internet.
3. What is the goal of site planning ?
4. List as many web browser you know

Self-Assessment Answer

The web environment

A Web environment is a conceptual space that allows a group of people to interact about common interests, tasks and ideas. Web environments take different forms depending on the tools being used, such as web logs (blogs) or e-mail lists. They are designed to support communication and collaboration for a community of non-technical users with software for sharing text, graphics, and photos. Some software is designed specifically to facilitate meetings, including the possibility of voice conversations coordinated with Web displays and live video.

Software for creating an environment usually includes a web site development function. The software allows the originator of the environment and participants to easily contribute new material or comments on existing material, including from a cell phone. The originator can set boundaries including controlled entry for a group, such as students in a class, or establish a space open to anyone and defined only by the topic. There are no geographical or time boundaries for participants in a web environment.

Key research Findings

(i) Students today are accustomed to working within networked environments. To a student in the 21st century a networked environment may include their home computer, cell phone, handheld, and personally designed Web sites (Tapscott, 1998).

(ii) Networked technology can enable teachers and students to build local and global communities that connect them with interested people and expand opportunities for teacher learning (Kozma, 2003).

(iii) The Web can be a key tool to fostering schoolwide change using technology (Sandholtz, Ringstaff, & Dwyer, 2000).

Implementation

The following strategies can help you begin the use of Web environments for teaching and learning.

i. Experiment with networked technologies. Use search engines to find and explore new Web services, social networking tools, and ideas for classroom use.

ii. Extend your professional reach. Use the Web to connect with other teachers and other classrooms around the world via weblogs and forums. Consider engaging with or setting up a weblog to create a private threaded discussion.

iii. Explore existing online systems. Free course management services allow you to set up discussion boards, create secure online assessment for students, and provide focused resources for your students.

iv. Build your own environment for parents and community members. New technologies can automatically send emails to notify parents of changes to schedules or provide easy access to your instructional expectations. Publish an online newsletter that enables feedback, inviting parents and the community to engage with your classroom activities.

v. Allow students to lead. Students are often the earliest adopters of new technologies. Allow students to use and create Web-based environments in learning activities.

Self-Assessment Question(s)

1. What are the benefitiaries of web environment
2. List the steps of using web environment

Self-Assessment Answer

1. - Web environment allows a group of people to interact about common interests, tasks and ideas.
   - Web environment support communication and collaboration for a community of non-technical users.
   - Web environment allows the originator to set boundaries including controlled entry for a group, such as students in a class.
2. - Experiment with networked technologies.
   - Extend your professional reach.

- Explore existing online systems.
- Build your own environment for parents and community members.
- Allow the community to lead.

Authoring Tool

Authoring tools are software applications provided to make the job of the web designer and programmer easier and like most other professions, a web designer uses some tools to bring the concept into frutiion.

Web authoring tools range from simple text editors like notepad, note++ to hing end graphic authoring tools and content mangagement systems like Adobe Dreamweaver, Adobe Photoshop, Wordpress and Joomla.

Now let's take a brief look at 3 examples Web Authoring tools:

Dreamweaver

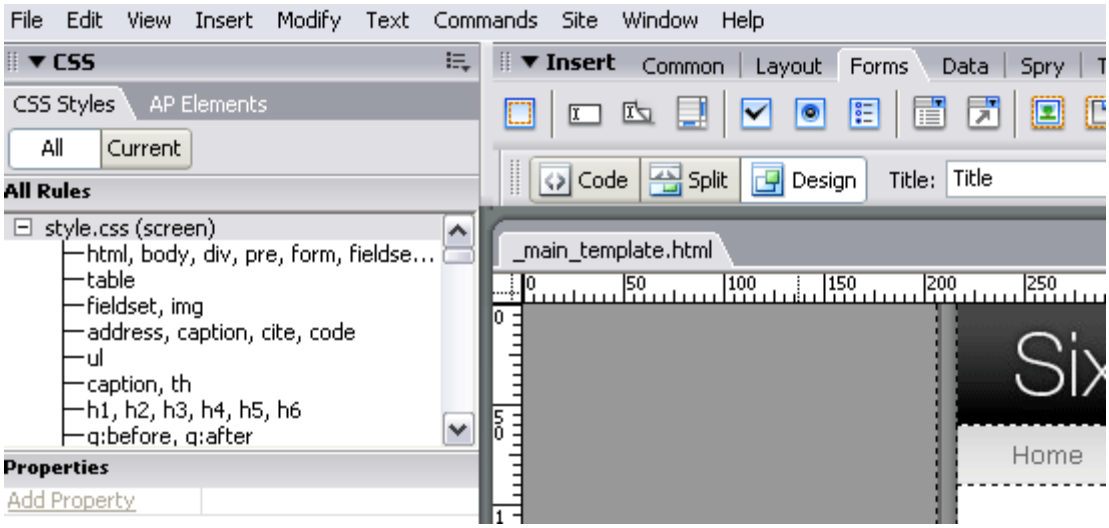


Fig. 3.3.1: Dreamweaver

Adobe Dreamweaver is a commercial application for web development that's available for the Mac and Windows operating systems. Its featured-packed suite of tools and options include: syntax highlighting and very smart Code Hinting, a built-in FTP client, project management and workflow options that make team work effortless, and Live View – which shows you a preview of your source code. Dreamweaver tightly integrates with other popular Adobe products such as Photoshop, allowing you to share Smart Objects for quick and easy updating and editing of graphics components

NOTEPAD++

Notepad++ is a free source code editor and Notepad replacement that supports several languages. Running in the MS Windows environment, its use is governed by GPL License.

Based on the powerful editing component Scintilla, Notepad++ is written in C++ and uses pure Win32 API and STL which ensures a higher execution speed and smaller program size. By optimizing as many routines as possible without losing user friendliness, Notepad++ is trying to reduce the world carbon dioxide emissions. When using less CPU power, the PC can throttle down and reduce power consumption, resulting in a greener environment.

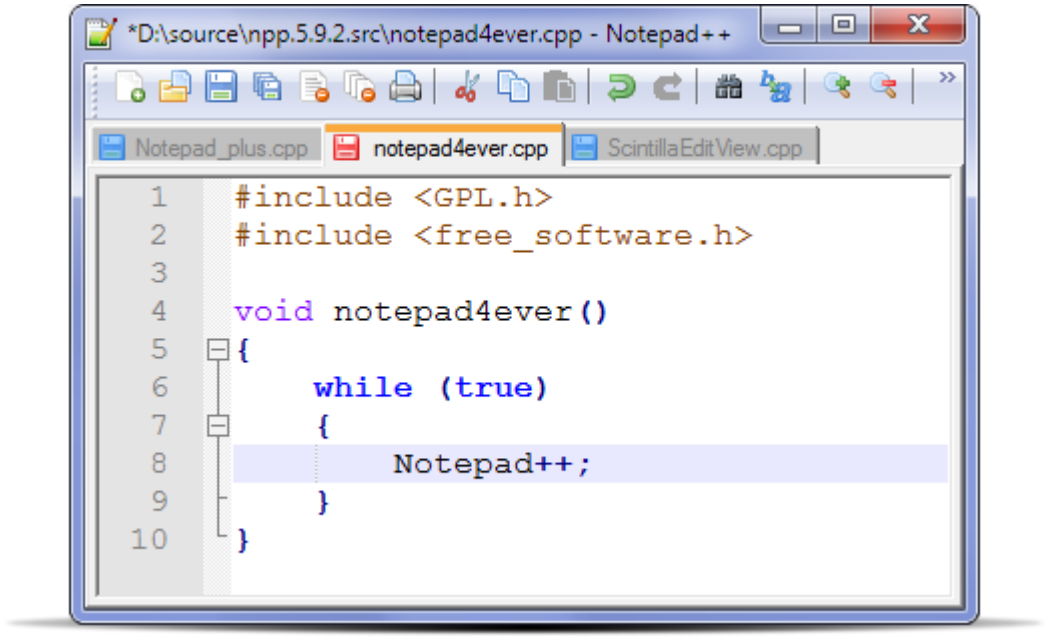


Fig. 3.3.2: Notepad++

WordPress

WordPress, content management system (CMS) developed in 2003 by American blogger Matt Mullenweg and British blogger Mike Little. WordPress is most often used to create blogs, but the program is sufficiently flexible that it can be used to create and design any sort of Web site. It is also an open-source product, so users can modify it for their own purposes.

Self-Assessment Question(s)

1. List the steps of using web environment

Self-Assessment Answer

4.0 Conclusion

Web server is the combination of computer and the program installed on it. Web server interacts with the client through a web browser. The internet is the interconnectivity of computer systems over a very large area for communication i.e sending and receiving of data in the form of text, images video and sound. A Web environment is a conceptual space that allows a group of people to interact about common interests, tasks and ideas. Authoring tools are software applications provided to make the job of the web designer and programmer easier and like most other professions, a web designer uses some tools to bring the concept into frution.

5.0 Summary

In this study unit we have been able to discuss the following topics

i. the history of internet
ii. the materials needed to connect to internet
iii. how to make better use of internet
iv. web server and its usefulness
v. web environment and its usefulness

6.0 Tutor Marked Assignments

1. Write short note on web environment
2. List at least 4 components needed for internet
3. What gave birth to the web standards ?.
4. Write short note on web server

7.0 References/Further Readings

Introduction to Web Standards http://dev.opera.com/articles/view/1-introduction-to-the-web-standards-cur/

History of the Internet http://dev.opera.com/articles/view/2-the-history-of-the-internet-and-the-w/

Responsive Web design http://www.smashingmagazine.com/responsive-web-design-guidelines-tutorials/

Unit 2

Concept of HTML

Content

1.0 Introduction

HTML or Hyper Text Markup Language is a set of set of codes or markup language that is used to write web pages. It simply describes a web page's content and its structure. HTML elements are in tags enclosed in a angle bracket e.g. <html>.

HTML is not case sensitive. This means that codes can be written in both capital letters e.g. <HTML> and in small letters e.g. <html>. Even with this it is recommended that your codes be written with small letters.

HTML forms the basic building blocks of all websites today. It also supports use of other web enabled languages such as Php, Css, Flash, Javascript, Java, Cold fusion and more all of with improve the quality of the web page.

2.0 Learning Outcomes

At the end of this unit, you should be able to:

i. Discuss the history of HTML
ii. Create a good web page using HTML
iii. Carry out basic editing on HTML
iv. Identify the HTML structural tags and use the tags correctly

3.0 Learning Content

3.1 HTML Overview

3.1.1 History of HTML

In 1980, physicist Tim Berners-Lee, who was a contractor at CERN, proposed and prototyped ENQUIRE, a system for CERN researchers to use and share documents. In 1989, Berners-Lee wrote a memo proposing an Internet-based hypertext system. Berners-Lee specified HTML and wrote the browser and server software in the last part of 1990. In that year, Berners-Lee and CERN data systems engineer Robert Cailliau collaborated on a joint request for funding, but the project was not formally adopted by CERN. In his personal notes from 1990 he lists some of the many areas in which hypertext is used" and puts an encyclopedia first.

Creating Your First Web Page

With what has been written so far, it's time for you to create your first web page. There are numerous tools that can be used to design web pages and web sites. These tools are:

i. Notepad
ii. Wordpad
iii. Note++

iv. Adobe Dreamweaver and so on

We would be using notepad. It's ok to use other too, but since we are focusing on the basics, I would recommend you tag along.

To find notepad on your Windows Operating system click on the Microsoft start button, click on the search bar and type notepad

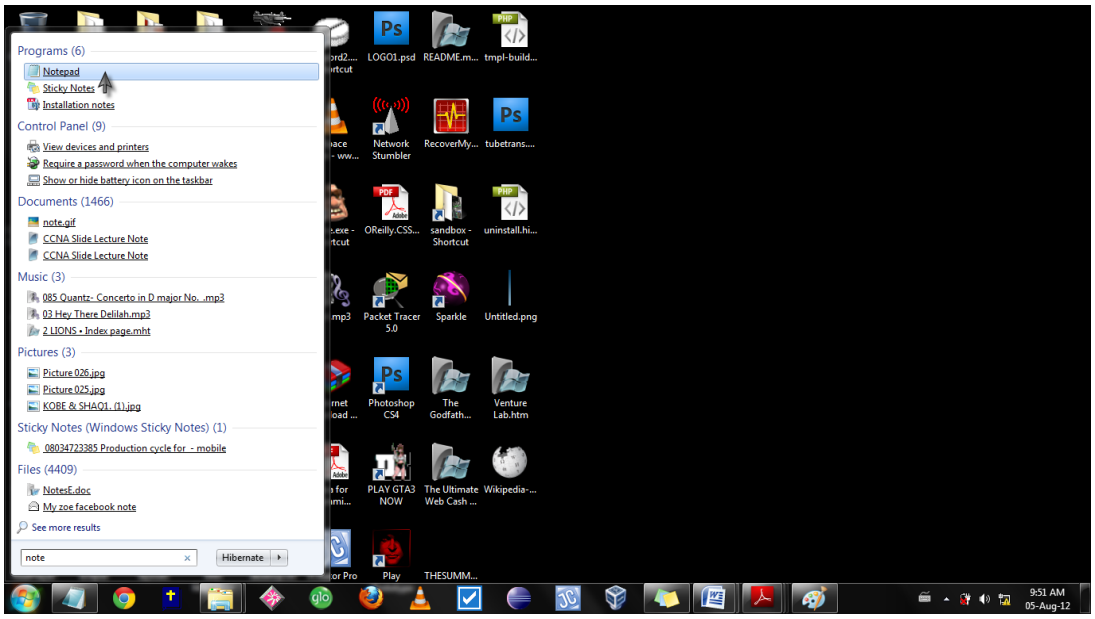


Fig. 3.1.1: Desktop Window

Once the notepad has been opened you can copy and paste or just type this

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Example page</title>
  </head>
  <body>
    <h1>Hello world</h1>
  </body>
</html>
```

To save the page, click on **file** then click on **save as,** then type the file name helloworld.html and then click **save.**

Note: *The page has to be saved as html or htm and not as text or .txt.*

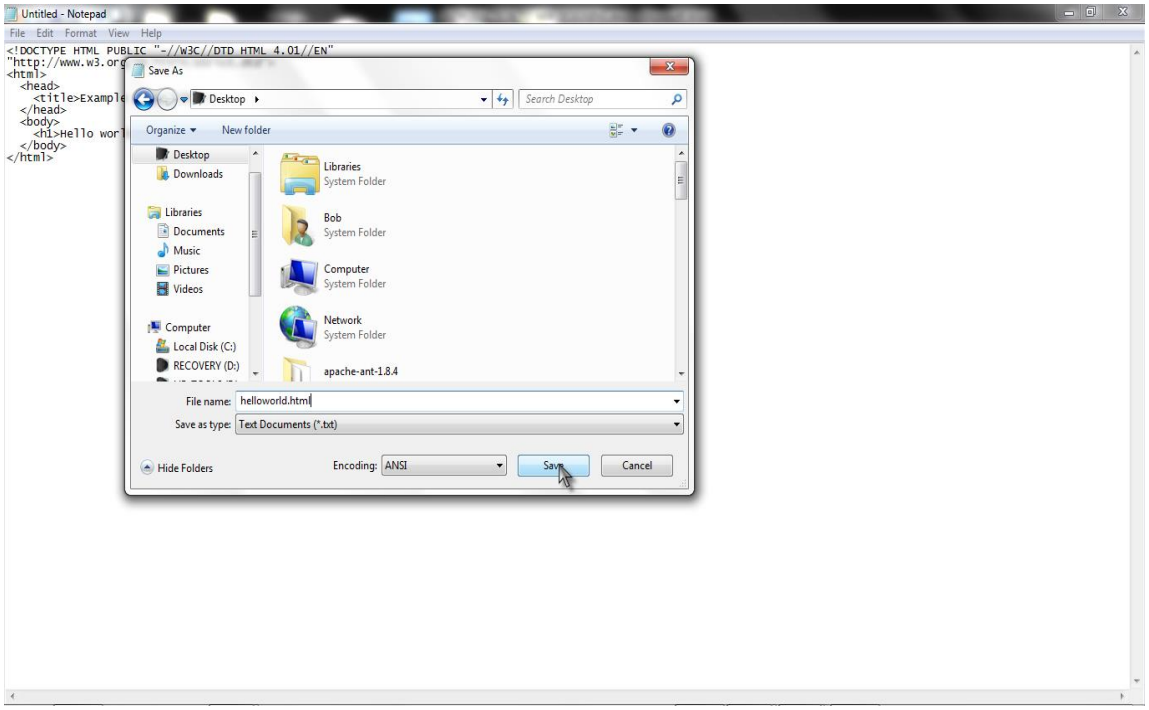


Fig. 3.1.2: Notepad Window

Now go to where you saved the page and double click it

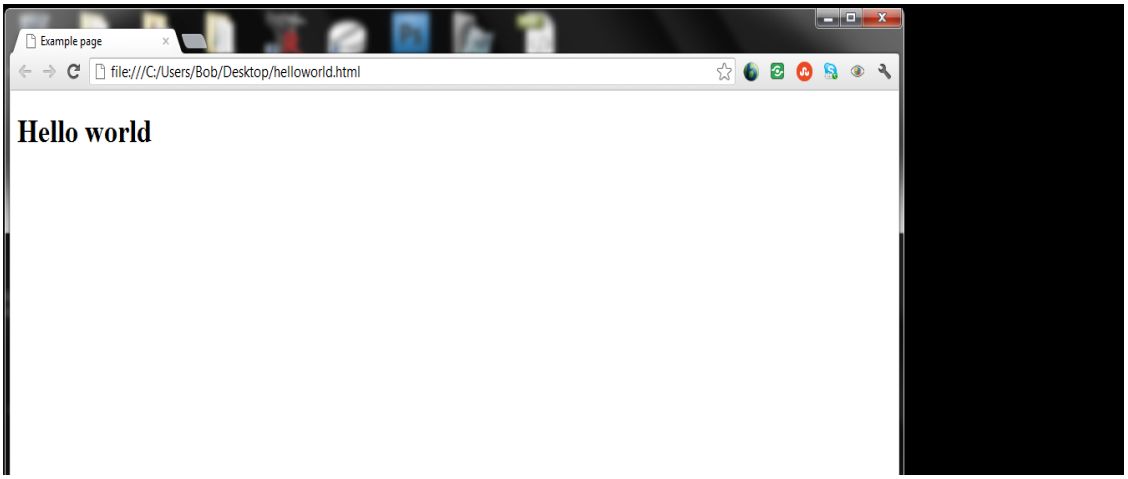


Fig. 3.1.3: Web Page

If your page looks like something that is above then, congratulation on designing your first web page.

If you want to edit that page then all you have to do is go back to where you saved the page, right-click on it, go open with and select notepad.

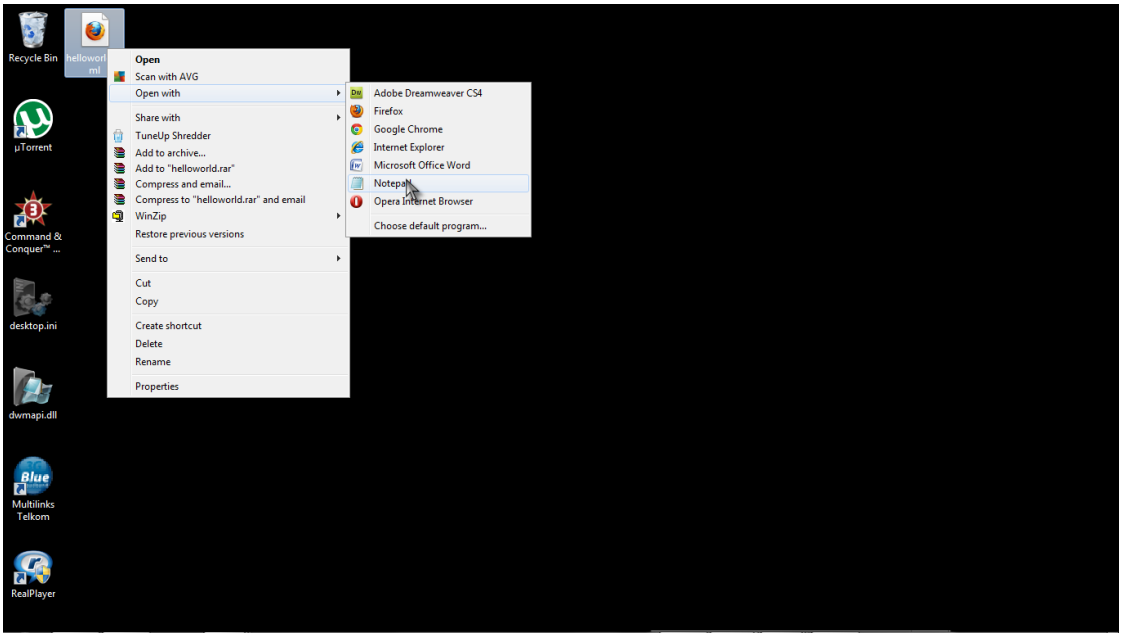


Fig. 3.1.4: Window Desktop

Basic Editing on HTML

How to space in HTML

Ok, let's try something with what we have so far

Create a new web page and instead of writing "Hello World" type

<h3>On the

first line</h3>

<h3>On the second line </h3>

Save the page and view it. When you open the web page what does it look like?

I am sure you have noticed that "On the first line" appeared on one single line instead of two lines, just as "On the second line"

This can be a source of confusion for first-time authors of an HTML document, who try to pad out text with extra spaces to achieve a desired indentation, or to get more spacing after the period between sentences and introduce more vertical space between paragraphs. Influencing the visual layout of your documents is not something to be done in the HTML source.

One way this can be done is by using the pre element <pre></pre>

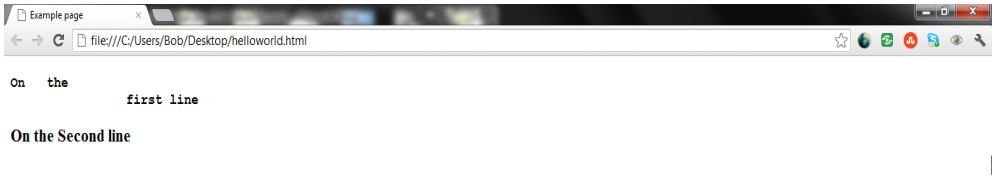
So now try this, rather than using <h3> tag alone add this <pre> like this

<h3><pre>On the

first line</pre></h3>

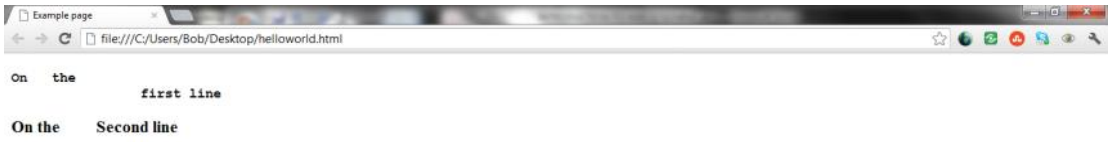
<h3>On the Second line</h3>

Save the page and view it again.



Another way of leaving just enough white space is to use the " " To test this go back to the previous web page we created and edit it again. Now let's edit the "<h3>on the second line</h3>" and add <h3>On the   Second line</h3>.

Save it and open the page to take a look at it again.



Self-Assessment Question(s)

1. What is HTML?
2. Mention 4 HTML tools.
3. Discuss how to creating space in HTML

Self-Assessment Answer

1. HTML or Hyper Text Markup Language is a set of set of codes or markup language that is used to write web pages. It simply describes a web page's content and its structure. HTML elements are in tags enclosed in a angle bracket e.g. <html>.
2. i. Notepad
   ii. WordPad
   iii. Note++
   iv. Adobe Dreamweaver

3.2 Structural HTML Tags

We could compare the structure HTML to that of a human body. For example the human body has a Head, Body and legs and so those HTML Documents. The smallest valid HTML document possible would be something like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Example page</title>
  </head>
  <body>
    <h1>Hello world</h1>
  </body>
</html>
```

The HTML Document could start with a document type element popularly known as doctype. The doctype describes the type of HTML that is being used.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html4/strict.dtd">

This enables the web browsers to interpret the document. It is not compulsory to include this however as said earlier it makes the job of the web browser easier.

Next is the first opening tag of the HTML element.

It is mandatory for every web page to have the HTML tag

<html>

Note that the HTML tab will also terminate last.

</html>

Like many other computer programming language once you open a tag, it has to be terminated at one point.

Inside the HTML element i.e. in between the <html> and </html>, there is the head element.

<head>

<title>Example page</title>

</head>

This is a wrapper to contain information about the document (the metadata). Inside the head is the title element, which defines the “Example page” heading in the menu bar of the web browser.

After the head tag is terminated i.e. </head>, there is a body element.

<body>

```
    <h1>Hello world</h1>
  </body>
```

The body wrapper i.e. <body></body> contains the actual content of the page. This is where text is displayed, colours are inserted, images are inserted and many more.

In this example, only a level-one header element <h1></h1>, which contains the text “Hello world.” And that’s our document in full.

As you can see, elements can also contain other elements. The body of the document will invariably end up involving many nested elements. Page divisions create the overall structure of the document, and will contain subdivisions. These will contain headings, paragraphs, lists and so on.

Self-Assessment Question(s)

1. List the elements of HTML structural tag

Self-Assessment Answer

HTML structure is made up of the several tags these include

- The HTML tag

  <html>

  </html>

  The HTML tag opens and ends the web page
- The Head tag

  <head></head>

  The head tag encapsulates the title tag e.g

  <head>

  <title></title>

  </head>
- The Title tag

  <title></title>

  The title tag encapsulates the name of the page.

  e.g.

  <head>

  <title>myfirstpage</title>

  </head>
- The Body tag

  <body></body>

  The body tag encapsulates the main content of the page. Without it the page will remain empty.

  <body>

  <h1>Hello world</h1>

```
</body>
```

4.0 Conclusion

HTML is a set of set of codes or markup language that is used to write web pages. For a web page to be designed the HTML code guidelines must be followed.

5.0 Summary

In this Study Unit, the following aspects have been discussed:

1. the history of HTML,
2. design of a good web page using HTML,
3. basic editing on HTML,
4. HTML structural tags and how to use them correctly.

6.0 Tutor Marked Assignments

1. Write briefly on the HTML
2. Discuss the structural HTML tags

7.0 References/Further Reading

1. Introduction to HTML http://www.w3schools.com/html/default.asp
2. Creating Cool Web Sites with HTML, XHTML, and CSS (2004)
3. A Beginner's Guide to HTML(1996)
4. Web Design Manual (2010)

Module 2

Formating, Link, Image and CSS

Unit 1: Formating Text, Creating Link And Adding Images In Html

Unit 2: Cascading Style Sheet And Server Side Includes

Unit 3: Graphics Gif

Unit 1

Formating Text, Creating Link and Adding Images in Html

Content

1.0 Introduction

A web page consist of text images, Audio and Video files that combine together into one page by using markup elements and attributes. We are going to be looking at how to create a good HTML document using tags that can be used to edit the attributes of the text.

2.0 Learning Outcomes

At the end of this Study Unit, you should be able to:

i. format text on the web page using HTML,

ii. create links on the web page,

iii. discuss the HTML menu tools,

iv. add image and other page elements like font, colour, etc,

v. do background editing features in HTML.

3.0 Learning Content

3.1 Formatting Text

Paragrahp Tags

The paragraph is the building block of most documents. In HTML a paragraph is represented by the p element, which takes no special attributes. For example:

 This is a very short paragraph. It only has two sentences.

A paragraph in many articles and books can contain just one sentence. Whilst the meaning (in terms of written prose) of “paragraph” is fairly clear, on the web much shorter areas of text are often wrapped in paragraph elements as the author believes this is more “semantic” than using a div element (we will cover these in a future article called “Generic containers”).

A paragraph is a collection of one or more sentences, just as in newspapers and books. On the web, it is good to use the paragraph element random piece of text in the page. If it is just a few words and not even a proper sentence, then it should probably not be marked up as a paragraph

Line Break

Another way to move to the next line of your HTML document is to use the break line element
</br>

For example we could write it like this

This is a new line</br>

But for you to really notice the line break element
</br> you need to have added previous content to the tag

Now based on the previous page we created, let's edit it by adding a few more lines and then adding the line break element into it:

This is a very short paragraph. It only has two sentences.

Let's just add more content to make this page a feel

busy.

This is a new line using break line or br</br>

This is another new line</br>

This is also another

Line</br>

This is another new line using paragraph element

This is another new line again using the same paragraph

Tag

Once you have saved the page and opened it on your browser it should look like this

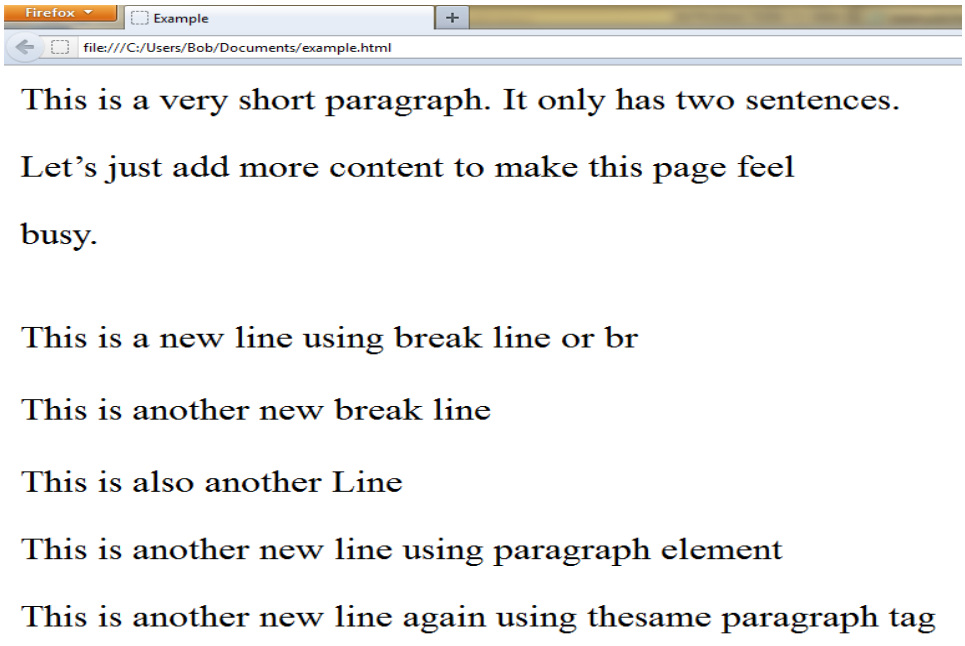


Fig. 3.1.1: Creating and edting paragraph

Did you notice the space between the paragraph "Let's just add more content to make this page feel busy" and the next line "This is a new break line br"

Break line is most suitable when you are about to start a new topic in your html document.

Headings

You don’t need to go far to see why you need headers, Open any book or any Newspaper or even any internet page. What you will see is a Bold Headline for example: **BREAKING NEWS!!!!!** The next line will have a smaller print. All these can be acheieved using header tags.

The six heading elements, H1 through H6, denote section headings

Headings have six different levels h1, h2, h3, h4, h5 and h6

h1 - can be used to specify level 1 headings. It can also be used appropirately to either describe a page or a section of a page, if the page has two distinct topics.

h2 - can be used to specify level 2 headings and can be used for as many as require h2 elements to denote sections on the page

h3 - can be used to specify level 3 headings and you can consider using H3 Elements for useful link groups to other relevant sources, or for information not immediately relevant to the page and so “3rd” in the pecking order

h4 – can be used for specifying level 4 headings.

h5 - can be used for specifying level 5 headings.

h6 - can be used for specifying level 6 headings you can used use it in sub links and footers.

Let’s apply this:

Open a new page and insert this into it

<h1>Welcome to my second website</h1>

<h2>Home Page</h2>

<h3> Thanks for calling in to my website. I hope your visit was worth it.<h3>

<h4>My name is Chris and I am a web developer in Lagos. </h4>

<h5> Thanks for dropping by! Feel free to join the discussion by leaving comments</h4>

<h6>For further information about me contact me on 123456789</h6>

<h6>Thanks</h6>

Once you have saved the page and opened it on your browser it should look like this

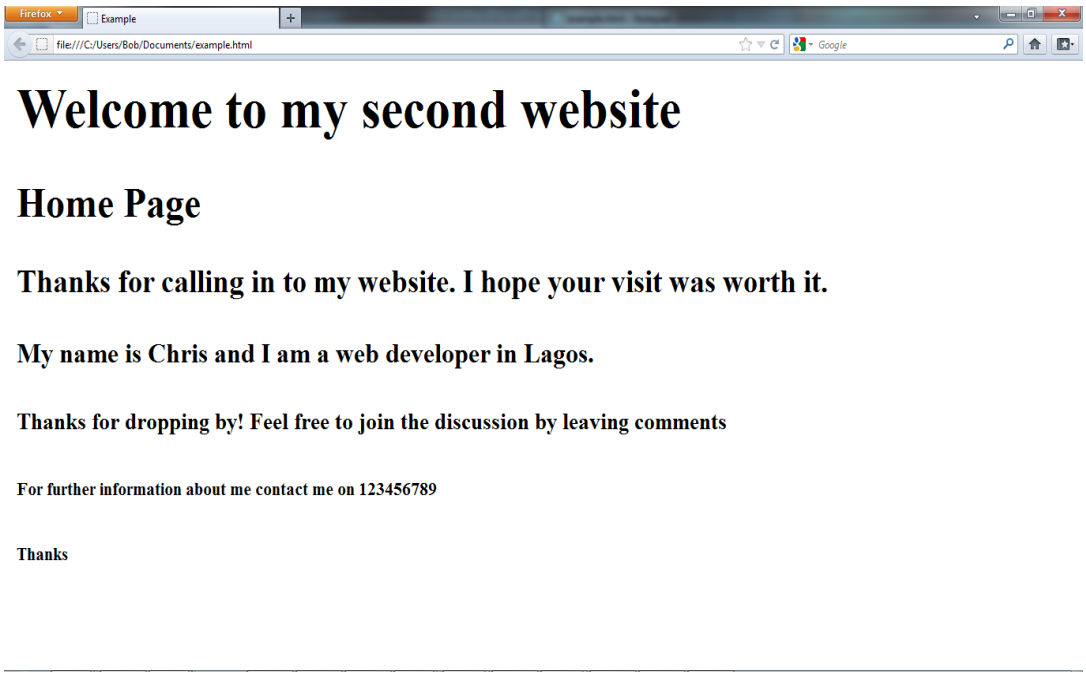


Fig.3.1.2 Creating Heading

Strong

Are you Strong or Bold?

Before showing you the strong elements and how to use it I want to quickly iron out a confusing issue.

There is a difference between using what are referred to as logical tags and tags that primarily affect visual layout. If you use a **<b>** tag to make the text **bold** it does look **bold**.

If you use a **** tag to make the text **bold** it also looks **bold** in most browsers.

Like every other element used previously, Strong element can be used by inserting the text inbetween the open and termination element

e.g Attension: All fingers are not equall

Now open a new page and insert the following inside

<h1>Welcome to my website</h1>

Home Page

 Thanks for calling in to my website. I hope your visit was worth it.
My name is Chris and I am a web developer in Lagos.   Thanks for dropping by! Feel free to join the discussion by leaving comments For further information about me contact me on 123456789</h6>
Thanks

Save the page and open it on your browser

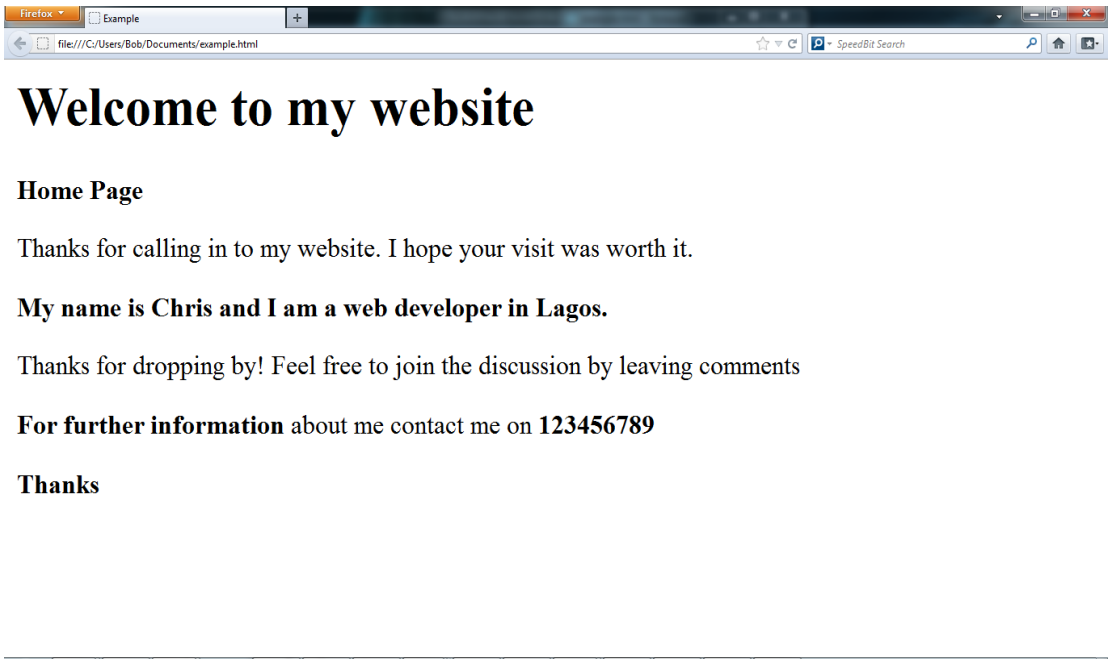


Fig. 3.1.3: Strong and Bold

Emphasis

Now let's take a look at how emphasis works in Web page design. The text is placed in between the Emphasis element

e.g This text is in emphasis

In the example above, all the text is in emphasis. Now Let's add some more features we have learnt earlier. In the example above, all the text is in emphasis. Now let's add some more features we have learnt earlier. Open a new web page and add this into it

<h1>Welcome to my site</h1>

Home Page

 Thanks for visiting my website.

I hope your visit was worth it.

Please note:

This text is in emphasis

Save it and open it with a browser it should look like this:

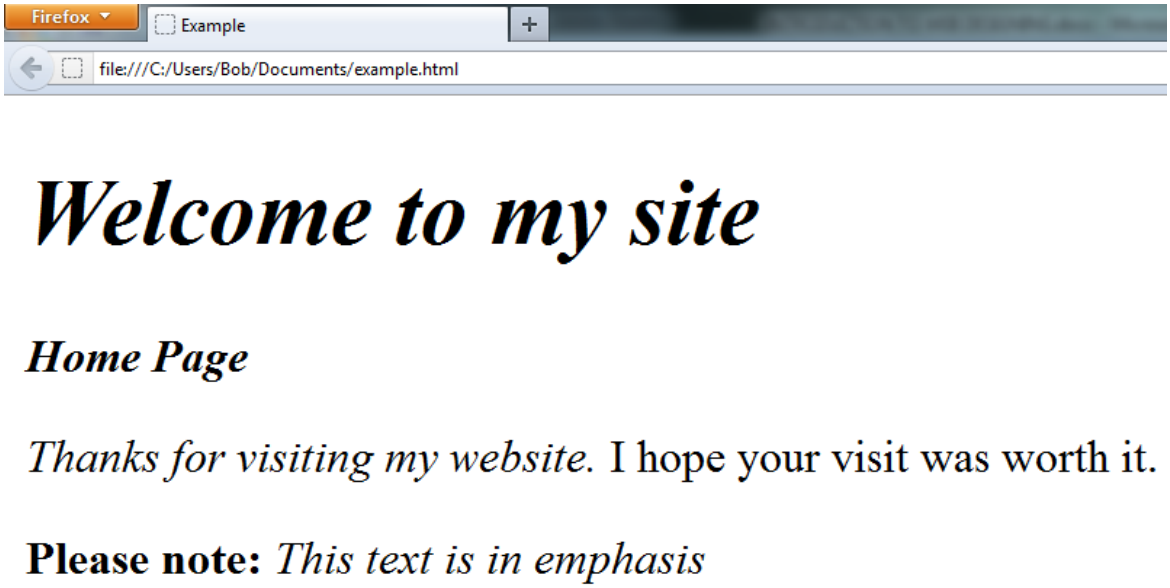


Fig. 3.1.4: Emphasis

Self-Assessment Question(s)

1. Discuss six (6) different levels of headings
2. Differentiate between strong and bold
3. Discuss the phrase “paragraph tags” in HTML

Self-Assessment Answer

1. h1 - can be used to specify level 1 headings. It can also be used appropirately to either describe a page or a section of a page, if the page has two distinct topics.

   h2 - can be used to specify level 2 headings and can be used for as many as require h2 elements to denote sections on the page

   h3 - can be used to specify level 3 headings and you can consider using H3 Elements for useful link groups to other relevant sources, or for information not immediately relevant to the page and so “3rd” in the pecking order

   h4 – can be used for specifying level 4 headings.

   h5 - can be used for specifying level 5 headings.

   h6 - can be used for specifying level 6 headings you can used use it in sub links and footers.

3.2 Creating Links

HTML menu tools—links, anchors and lists

There are several different types of menu and navigation idioms to consider in HTML, all connected closely with link and a (anchor) elements. In a nutshell:

link elements describe relationships across several documents. You can for example tell a user agent that the current document is part of a larger course that spans several documents and what other document is the table of contents.

Anchors (aka a elements) allow you to either link to another document or to a certain section of the current document. These don't get automatically followed by the user agent; instead they'll be activated by your visitors by whatever mean available to them (mouse, keyboard, voice recognition, switch access…)

If you haven't read the links article and lists article earlier in the course, I'd like you to go back and have a read as I build on some of the information given there to avoid repetition.

Anchors/links do not just become menus on their own—you need to structure and style them to let both the browser and your users know that their function is as a navigation menu, not just a set of random links. If the order of the pages is not important you can use an unordered list.

Note that I've put an id on the ul elements. The reason for that is to provide a hook for styling it with CSS and adding special behaviour with JavaScript later on. An id is a very inexpensive way to allow other technologies to single out a certain element in HTML.

If the order in which the visitors go through all the documents is important then you need to use an ordered list. If for example you have a multi-document online course where one tutorial builds on top of the last one, you could use an ordered list like in this ordered list example.

Using lists to create menus is great for several reasons:

All the HTML is contained in a single list element (ul for example), which means you can use the cascade in CSS to style each differently and it is easy to reach the elements in JavaScript going from the top level down.

Lists can be nested, which means you can easily create several levels of navigation hierarchy.

Even without any styling applied to the list, the browser rendering of the HTML makes sense and it is easy to grasp for a visitor that these links belong together and make up one logical unit.

You nest lists by embedding the nested list inside the li element, not after it.

Notice that browsers display both examples in the same way. Browser display should never be an indicator for the quality of your code. An invalid HTML construct like the

wrong example shown on the above example page will be hard to style, add behaviour to with JavaScript or convert to another format. The structure of nested ULs should always be and never

.

The need for flexibility

The menu of a site is unlikely to stay the same for very long - sites tend to grow organically as functionality is added and the user base grows, so you should create menus with scope for menu items to be added and removed as the site progresses, and for menus to be translated into different languages (so links will change in length). Also, you may well find yourself working on sites where the HTML for menus is created dynamically using server-side languages rather than with static HTML. This does not however mean that knowing HTML will become obsolete; it will actually become more important as this knowledge will still be needed to create HTML templates for the server-side script to work with.

Different types of menu

There are several types of menus you will be called upon to create in HTML, as you work on different web sites. Most of these can be created with lists, although sometimes interface restrictions force you to go another route (more on that later). The list-based menus you will be likely to create are as follows:

In-page navigation: for example a table of contents for a single page, with links pointing to the different sections on the page.

Site navigation: a navigation bar for your whole web site (or a subsection of it), with links pointing to different pages on the same site.

Content-contextual navigation: a list of links that point to pages closely related to the subject of the page you're currently on, either on the same site, or different ones.

Sitemaps: large lists of links that point to all the different pages of a web site, grouped into related sublists to make them easier to make sense of.

Pagination: links pointing to other pages that make up further sections or parts of a whole, along with the current page, for example part 1, part 2, and part 3 of an article.

Site navigation

Site navigation is most probably the most common menu type you'll need to create. It is the menu of the whole site (or a subset of it), showing both the options visitors can choose from and the hierarchy of the site. Lists are perfect for this purpose, as you'll see from this site navigation example.

There aren't many surprises here, at least not from a pure HTML point of view. Later on in the course we'll talk about styling these kind of menus with CSS and adding behaviour via JavaScript. One important thing to consider is how to highlight the

current document in the menu, to give the user a sense of being in a particular place, and that they are moving location (even though in reality they aren't, unless of course they are using a mobile device to browse the Web!). This is what I'll look at next.

Providing visitors with a "You are here" feeling

One golden rule of web development and navigation is that the current document should never link to itself but instead be obviously different to the other entries in the menu. This is important as it gives the visitors something to hold on to and tells them where they are on their journey through your site. There are edge cases like web applications, permalinks in blogs and so called "one page" web sites but in 99% of cases a link to the document you are already looking at is redundant and confusing to the visitor.

In the links tutorial, I stated that a link is an agreement and a liability: you offer visitors a way to reach more information that they need, but you need to be careful—you'll lose credibility and trust if that link doesn't give the users what they want, and/or results in unexpected behaviour. If you offer for example a link that points to the current document, activating it will reload the document. As a user this is something you don't expect - what purpose did clicking this link have? This results in users getting confused.

How many options should you give users at one time?

Another issue to consider is how many options you want to give visitors. A lot of menus you see on the web try to make sure that every page in the site can be accessed from one single menu. This is where scripting and CSS trickery comes in - you can make the menu more manageable by hiding parts of the menu until the users select certain areas (rollover menus, as they are sometimes called). This is clever from a technical point of view, but there are several issues with this approach:

1. Not all visitors will be able to use the clever trick as intended; keyboard users for example will have to tab through all links on the page just to reach the one they are looking for.
2. You need to add a lot of HTML to each document of your site to achieve this, and a lot of it can be redundant on many pages. If I drill down three levels in your menu to reach a document I want to read, I don't need to see options leading me to 4, 5, and 6 levels deep.
3. You can overwhelm visitors if you present them with too many options at once - humans don't like making decisions. Think about how long it can take you to pick a meal from a lengthy restaurant menu.
4. If there is not much content on a page but a lot of links, search engines will assume that there is not much valid information on this page and not give the page much attention, so it is harder to find when searching the Web.
5. All in all, it is up to you how many items you put into a menu - different designs will call for different choices - but if in doubt, you should try cutting your menus down

to only the links to the main sections of the site. You can always provide further submenus where appropriate.

Contextual menus

Contextual menus are links that build on the content of the current document and offer more information related to the current page you are on. A classic example is the “related articles” links you tend to get at the bottom of news articles, as shown in Figure 1.

An example of a contextual menu—a news article offering related news items at the bottom.

This is a slightly different thing to context menus in software user interfaces, which offer different options depending on where they are accessed (like the right-click or Ctrl + click menus you find in desktop applications that offer specific options depending on where your mouse pointer is at the time).

Contextual menus on web sites are a great way to promote content on other parts of the site; in terms of HTML they are just another list of links.

Sitemaps

Sitemaps are what you might expect - maps of all the different pages (or the main sections if you are talking about really huge sites) of your site. They allow your site’s visitors to get a glimpse of the overall structure of your site, and go anywhere they need to fast-even if the page they need is deep within your page hierarchy.

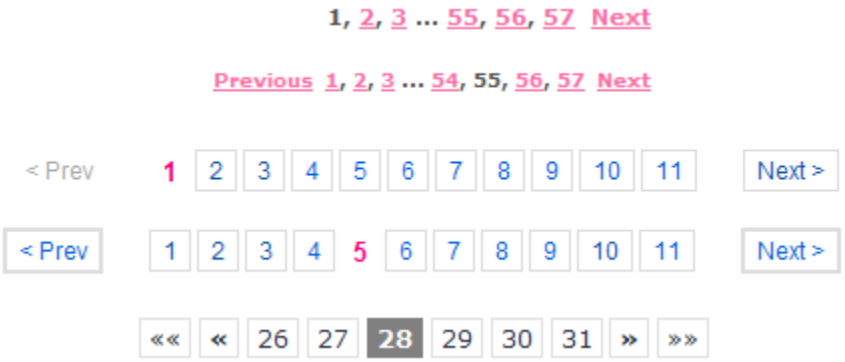
Both sitemaps and site searches are a great way of offering visitors a fallback option when they are lost or to offer quick access for those who are in a hurry.

From an HTML point of view they could either be one massive nested list full of links or - in the case of very large sites - section headings with nested links of section-specific hierarchies or even search forms for each of the sections.

Pagination

Pagination is necessary when you have to offer a way to navigate through large documents split into separate pages. You'll find pagination for example in large image archives or search result pages (like Google or Yahoo search.)

Pagination is different from other types of navigation because it does normally link back to the same document—but with a link that has more information in it like which page to start from. Some examples of pagination are shown in Figure 2:



Pagination menus allow visitors to go through large sets of data without losing track of where they are.

The HTML is nothing ground-breaking—once again you offer a list of links with the current link (indicating which chunk of data is shown and how far down in your pagination you are) not being linked and highlighted (eg with a strong element).

The main difference to site navigation is that there is a lot of programming logic going on with pagination. Depending on where you are in the whole data set you need to show or hide the previous, next, first and last links. If you have really massive amounts of information to navigate through you also want to offer links to landmarks like 100, 200 and many more options. This is why you are not very likely to hard-code menus like these in HTML but instead create them on the server-side. This does not change the rules however - the current chunk should not link to itself and you shouldn't offer links that lead nowhere.

When lists are not enough—image maps and forms

In 99% of the cases the ordered or unordered list is a sufficient HTML construct for menus, especially as the logical order and nesting also allows for styling with CSS very nicely. There are however some situations that may require different design techniques.

Setting hotspots with image maps

One technique is client side image maps. Image maps turn an image into a menu by turning sections of the images into interactive areas that you can link to different documents. The image map example associated with this section turns an image into a clickable menu.

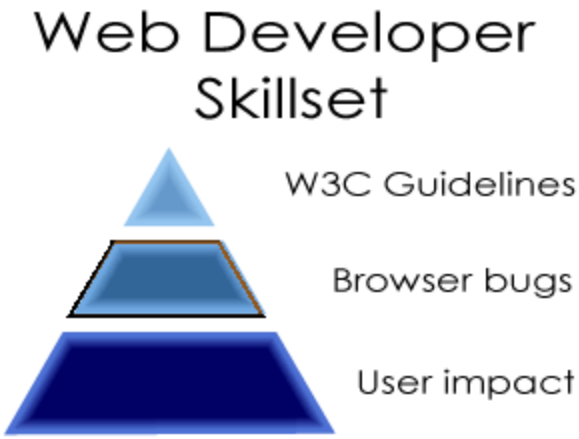


Fig. 3.2.1: Web Developer Skillset

By defining a map with area elements you can turn parts of an image into interactive elements. You can turn any image into a menu by defining a map with different areas (also called hotspots). You give the map a name attribute and connect the image and the map using the use map attribute on the image element. Notice that this works exactly like in-page links, which means that you need to precede the value of the use map attribute with a hash.

Each area should have several attributes:

href

defines the URL the area should link to (which could also be a target in the same document)

alt

defines alternative text in case the image can not be found or the user agent does not support images

shape

defines the shape of the area. This can be rect for rectangles, circle for circles or poly for irregular shapes defined using polygons.

coords

defines the coordinates in the image that should become hotspots—these values are measured from the top left corner of the image, and can be measured in pixels or percentages. For rectangular areas you only need to define the top left and the bottom right corner; for circles you need to define the center of the circle and the radius; for polygons you need to provide a list of all the corner points.

Image maps are not much fun to define and type in as HTML, which is why image manipulation tools like Adobe Image Ready or Fireworks offer an option to create them visually (they generate the HTML for you).

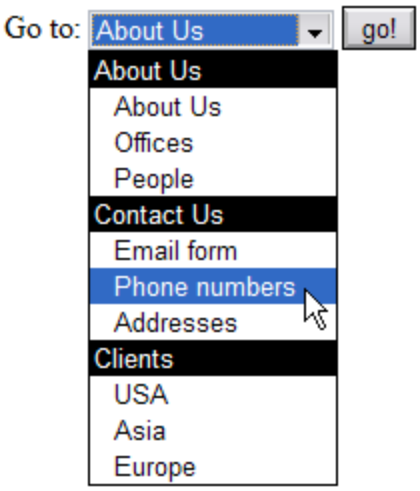


Fig. 3.2.2: Option Button

Select menus can get option groups that allow you to tell visitors which options belong together. This is the rendering on Opera 9.5.

This technique has the benefit of using up hardly any space but it also means that you need to have a server-side script to send the visitors to the pages they have chosen. You can also use JavaScript to make the links work, but you cannot rely on JavaScript being available - you need to make sure your users can still make use of the menu with JavaScript disabled.

Another, less obvious benefit is that you don't offer too many links in the same document. This means that you don't overwhelm users of assistive technologies (who often tend to be presented with the links in one big list). It also means that search engines don't consider the links on your page worthless as the link to text ratio makes the document appear to be a sitemap. However, many assistive technologies can produce a map of your pages' links; if your important links are all in a select menu, there is a chance that a visitor might never chance upon them. It is therefore a good idea to offer anchor links to the main destination pages and select element menus to offer more options. Visitors will be able to use them, but machines like search engine robots don't need to know they exist.

Where to put the menu

One last thing to mention about HTML menus is that the placement of the menu plays a large role. Consider visitors that have no scrolling mechanism or might not be able to see so rely on keyboard navigation to find their way around your site. The first thing they'll encounter when they load the document is its location and the title; next the document gets read top to bottom, stopping at each link to ask the visitor if they wanted to follow this link or not. Other options are to get a list of all the links or to jump from heading to heading.

If the menu is at the top of document, it will be the first thing the user will meet. Having to skip through 15 or 20 links before they get to any content could get really annoying.

There are two workarounds available. First, you could put the menu after the main content of the document (you can still place it at the top the screen using CSS if you wish). Second, you could offer a skip link. Skip links are simply links placed before the main menu that link to the start of the content, allowing the visitor to skip over the menu and get to the content immediately if they wish. You can add another “go to menu” link at the end of the document to make it easy to get back up to the top.

Links are the doors that connect pages together and makes it a web site.

Link elements<href></href> can be used as shown below:

<href=”address”>The name of the link</href>

We are going to create three simple pages using what we have learnt so far and are going to create some links connecting these pages together.

Open a new page and add these codes, I delibrately highlighted the important codes so that you see how to create links.

PAGE 1

body bgcolor="teal" background="bg.jpg">

<a href = "example.html">Home</a href> <a href = "example2.html">My Holiday Pictures</a href> <a href = "example3.html">About us</a href>

<h1>Welcome to my website</h1>

Lorem ipsum dolor sit amet,

consectetuer adipiscing elit.

Praesent aliquam, justo

convallis luctus rutrum,

erat nulla fermentum diam,

at nonummy quam ante ac quam.

Maecenas urna purus, fermentum

id, molestie in, commodo porttitor,

felis. Nam blandit quam ut lacus.

PAGE 2

<body bgcolor="teal" >

<a href = "example.html">Page 1</a href>

** <a href = "example2.html">Page 2</a href> **

** <a href = "example3.html">Page 3</a href> **

<h1> MY HOLIDAY PICTURES!!!! </h1>

Here is a picture of some daisies i took on the road

</br>

Some nice grass!!! I just had to take the picture

</br>

On the road again!!!

PAGE 3

<body bgcolor="grey" >

<a href = "example.html">Page 1</a href> ~ <a href = "example2.html">Page 2</a href> ~ <a href = "example3.html">Page 3</a href>

<font face = "Lucida Sans Unicode">
<h1>ABOUT US</h1>
Lorem ipsum dolor sit amet,
consectetuer adipiscing elit.
Praesent aliquam, justo
convallis luctus rutrum,
erat nulla fermentum diam,
at nonummy quam ante ac quam.
Maecenas urna purus, fermentum
id, molestie in, commodo porttitor,
felis. Nam blandit quam ut lacus.

Save these pages, it should look like this

PAGE 1 (HOME)

Fig. 3.2.3: Home Page

PAGE 2 (My Holiday Pictures)

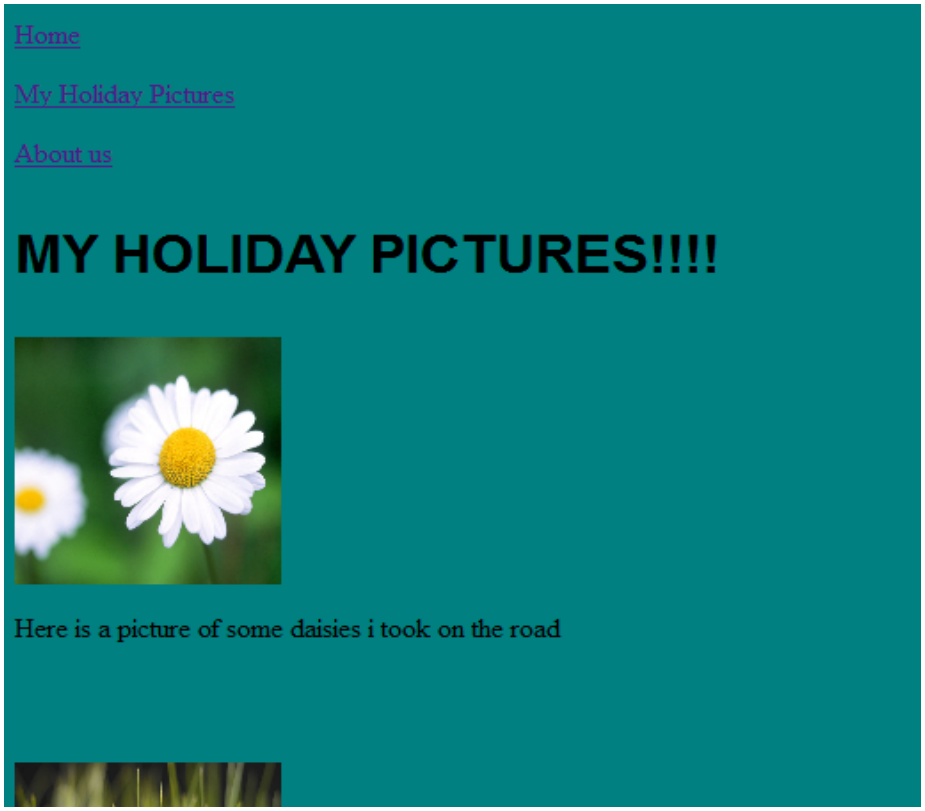


Fig. 3.2.4: Picture

PAGE 3 (ABOUT US)

Fig.3.2.5: About Us

Self-Assessment Question(s)

1. Discuss these following briefly in your own words;
   a. Site navigation.
   b. Page navigation.
   c. Link and anchors.

Self-Assessment Answer

a. Site navigation is the most common menu type you'll need to create. It is the menu of the whole site (or a subset of it), showing both the options visitors can choose from and the hierarchy of the site
b. Pagination is when you offer a way to navigate through large documents by splitting it into separate pages. Pagination is used in large image archives or search result pages (like Google or Yahoo search.)
   Pagination is different from other types of navigation because it does normally link back to the same document
c. Links are the doors that connect pages together and makes it a web site.

Link elements<href></href> can be used as shown below:

<href=”address”>The name of the link</href> for example <href= “welcome.html”>Welcome</href>

3.3 Adding Images and other page elements

Previously you learnt how to insert background colours and images to our web page. I remember reading books when i was young, I hated reading books that did not have images in them. Without Images the page will look dull, and unattractive. Two third of the time Images keep people on the page and entertain them.

You can use the Image element to insert the image.

So let's insert an image into the page that we had previously made

```
<body bgcolor="teal" background="bg.jpg">
<font face = "Arial"><h1><p>MY HOLIDAY PICTURES!!!!</p></h1></font>
<p><img src="Daisy.jpg" /></p>
<p>Here are some pictures of some daisies i took on the road</p>
<br></br>
<p><img src="Keynote Grass Widescreen.jpg" /></p>
<p>Some nice grass!!! I just had to take the picture</p>
<br></br>
```


On the road again!!!</p

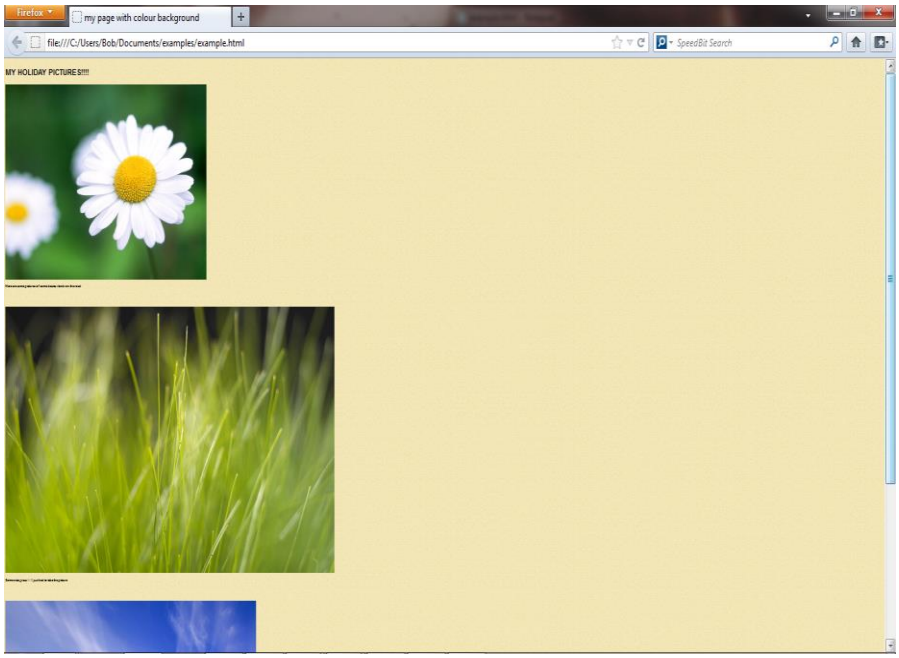


Fig. 3.3.1: Images

Blast!!! The images are either too big or too small, what do you do about that? That's simple, let's set the images to a fixed height and width.

****

<body bgcolor="teal" background="bg.jpg">

<h1> MY HOLIDAY PICTURES!!!! </h1>

Here is a picture of some daisies i took on the road

</br>

Some nice grass!!! I just had to take the picture

</br>

On the road again!!!

Once you have saved the page and opened it on your browser it should look like this.

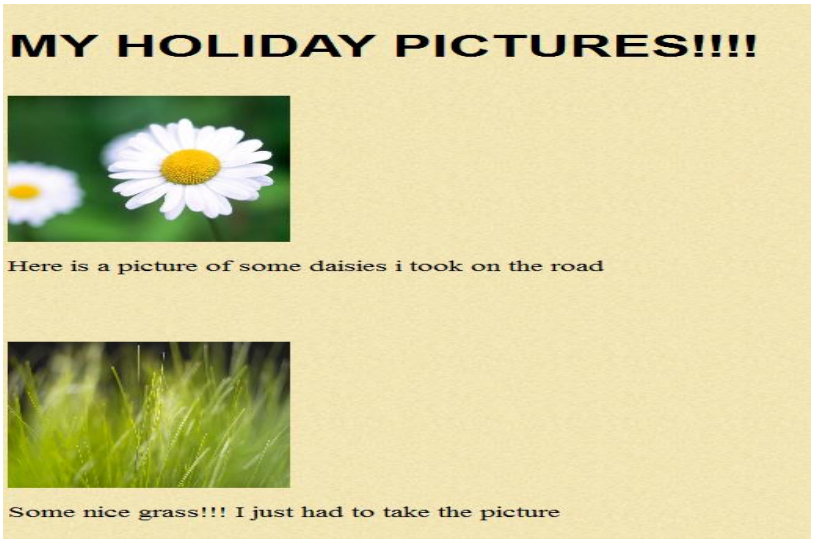


Fig. 3.1.2: Pictures

3.4 Tables, Frames, Forms, Specifying Colour in HTML,

Tables are an important addition to HTML that originated in the development labs at Netscape Communications Corporation. Unlike the tables in word processor, however, HTML tables can be quite compelling. You may even find yourself naturally boxing up groups of icons, taking a list of bullet items, and making a table out of them, or who knows what else! If you want to have material adjacent on a page, perhaps multiple columns of text, tables are unquestionably your best bet.

Fundamentally, tables are composed of data cells and organized into rows, the collection of which is called a *table*. In HTML, table data cells are denoted by <td> and </td>. These cells are collected neatly into rows with <tr> and </tr>, and the table itself starts with <table> and ends, logically enough, with </table>.

Basically you may have to specify allot of parameters for each tow and make sure that each cell is surrounded by <td></td>. It may feel challenging at first but you will get used to it. Let's create one

Take a look at these codes to create a list of fruits

```
<font colour="brown"><h2>List of Fruit</h2></font>
<table border="1">
<tr>
<td>Apples</td>
<td>Oranges</td>
<td>Pineapples</td>
<td>Mangoes</td>
</tr>
</table>
```

Save the page and view it again it should like this

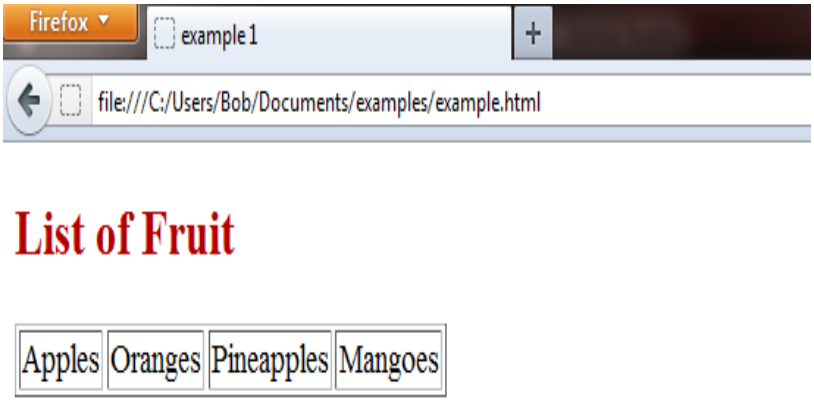


Fig. 3.4.1:

Now that you have a horizontal table how about making this table vertical? It's simple, take a look at these codes

```
<font color="brown"><h2>List of Fruit</h2></font>
<table border="1">
<tr>
<td>Apples</td>
</tr>
<tr>
<td>Oranges</td>
</tr>
<tr>
<td>Pineapples</td>
</tr>
<tr>
<td>Mangoes</td>
</tr>
</table>
```

Save it and open it on your browser. It should appear like this:

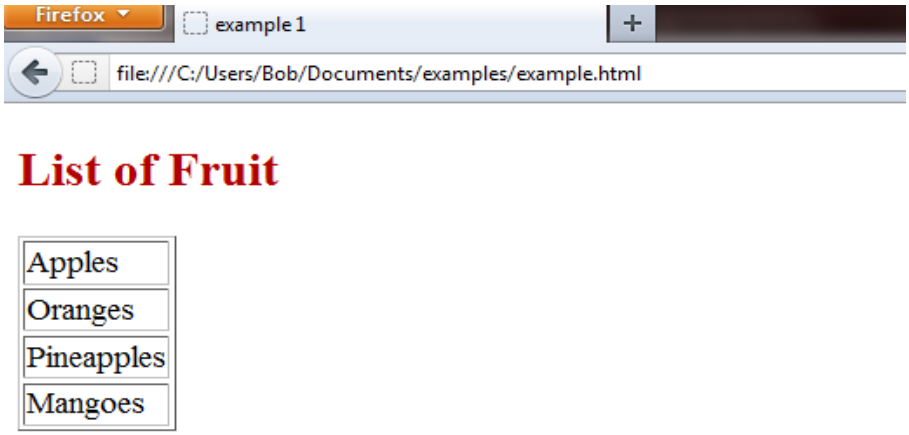
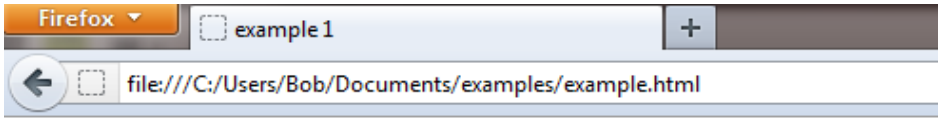


Fig. 3.4.2

One more thing, what if we want to stretch the table and at the same time
What we will get is these

```
<font color="brown"><h2>List of Fruit</h2></font>
<table border="1" width="358">
<tr>
<td>Fruits</td>
<td>Prices</td>
</tr>
<tr>
<td>Apples</td>
<td>₦50</td>
</tr>
<tr>
<td>Oranges</td>
<td>₦30</td>
</tr>
<tr>
<td>Pineapples</td>
<td>₦350</td>
</tr>
<tr>
<td>Mangoes</td>
<td>₦20</td>
</tr>
</table>
```

Save it and open it your browser. It should appear like this



List of Fruit

Fruits	Prices
Apples	N50
Oranges	N30
Pineapples	N350
Mangoes	N20

Fig. 3.4.3

How to design layout using tables

So far, you have learnt how to edit texts insert images and backgrounds. But not withstanding all we have learnt our page still looks disorganized and less attractive.

Now that you have created a table, let's see how you can use it to create a layout to organise your page and make it look cool.

In order for you to accomplish this Open a new page and insert these codes. There are many appraches to getting this effect but let's focus on one. We will be creating one table and three columns.

One column for the Sidebar1, One column for the Main Content and the last one for Sidebar2 Content.

Don't worry it's not what we have not done before. Just copy these codes and we can get this done.

```
<body bgcolor="#0099FF">
<h1>Welcome to my site</h1>
<table border="1" width = "1000" height="500">
<tr>
<td width="200" valign="top" bgcolor="#FFFFFF" >
<h3>Sidebar1 Content</h3>
<p>Lorem ipsum dolor sit amet,</p>
<p>consectetuer adipiscing elit.</p>
<p>Praesent aliquam, justo</p>
<p>convallis luctus rutrum,</p>
<p>erat nulla fermentum diam,</p>
<p>at nonummy quam ante ac quam.</p>
<p>Maecenas urna purus, fermentum</p>
<p>id, molestie in, commodo porttitor,</p>
<p>felis. Nam blandit quam ut lacus.</p>
</td>
<td valign="top" bgcolor="#FFFFFF" ><h1>Main Content</h1>
<p>Lorem ipsum dolor sit amet,consectetuer adipiscing elit. Praesent aliquam, justo</p>
<p>convallis luctus rutrum, erat nulla fermentum diam, at nonummy quam ante ac quam.</p>
<p>Maecenas urna purus, fermentum id, molestie in, commodo porttitor,</p>
<p>felis. Nam blandit quam ut lacus.</p>
<p>Lorem ipsum dolor sit amet,consectetuer adipiscing elit. Praesent aliquam, justo</p>
<p>convallis luctus rutrum, erat nulla fermentum diam, at nonummy quam ante ac quam.</p>
<p>Maecenas urna purus, fermentum id, molestie in, commodo porttitor,</p>
<p>felis. Nam blandit quam ut lacus.</p></td>
<td valign="top" width="200" bgcolor="#FFFFFF" >
<h3>Sidebar2 Content</h3>
```

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit.
Praesent aliquam, justo
convallis luctus rutrum,
erat nulla fermentum diam,
at nonummy quam ante ac quam.
Maecenas urna purus, fermentum
id, molestie in, commodo porttitor,
felis. Nam blandit quam ut lacus.
</td>
</tr>
</table>

Save it and it should look like this:

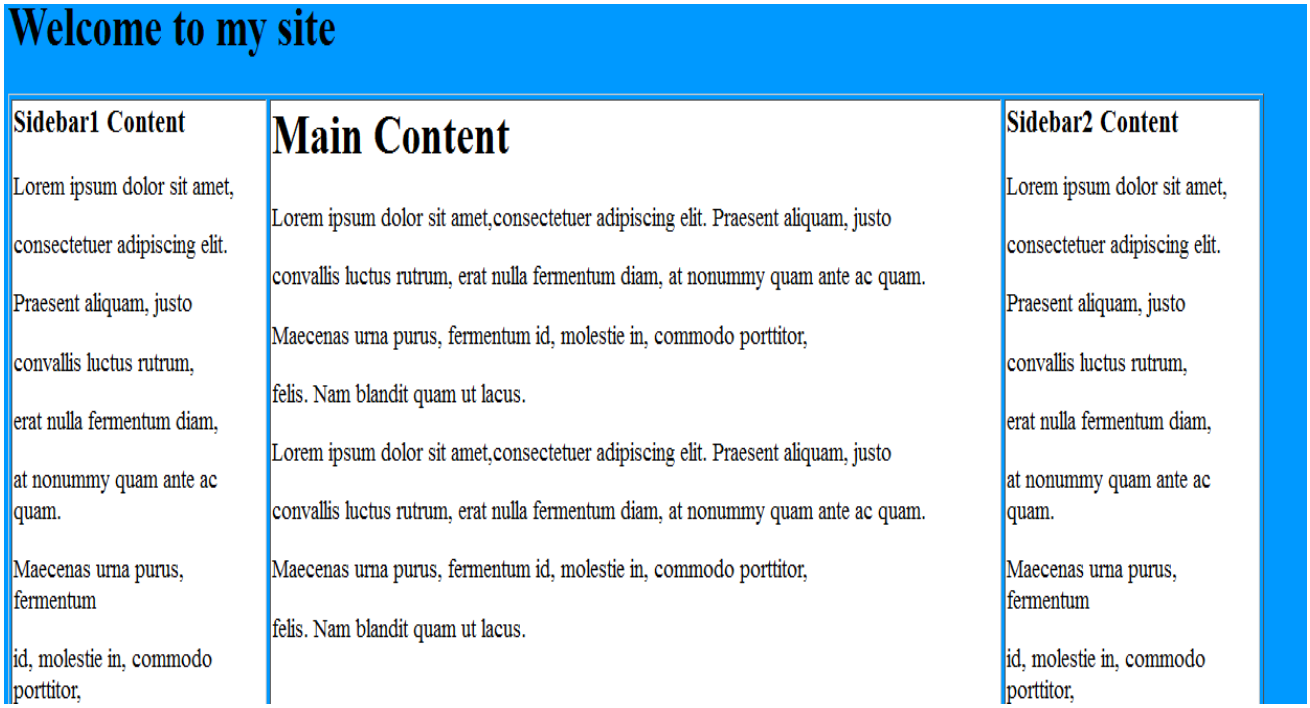


Fig. 3.4.4

Let's change the Sidebar1 Content into links, simply edit the content and add these instead:

<td width="200" bgcolor="#FFFFFF" valign="top" >

Home

My Holiday Picture

About us

</td>

Form

Everyone has seen a form. Everyone has used one. But have you coded one?

A form online is any area where you can input information into a web page, for example entering text or numbers into a text box, checking a tick box, causing a radio button to "fill in", or selecting an option from a list. The form is then submitted to the web site when you push the submit button.

You'll see forms used on the Web everywhere, for entering user names and passwords on login screens, commenting on blogs, filling your profile in on a social networking site, or specifying billing information on a shopping site.

It is easy to create a form, but what about a web standards compliant form? By now, if you have been working through the Opera Web Standards Curriculum, you are hopefully convinced that web standards are the way to proceed forward. The code required to create standards compliant accessible form are no more work to implement than a sloppy form.

So, let's start with the most basic and simple form that one could possibly want to use and build our way up in complexity after that—in this article I'll cover all the basics you need to know to create elegant, accessible form structures with HTML. The article structure is as follows:

Step one: The basic code

Let's start off with a really basic comment form, the sort of form you would use on a web site to allow people to give you feedback on something such as an article you've written, or allow someone to contact you without knowing your e-mail address. The code looks like this:

<form>

Name: <input type="text" name="name" id="name" value="" />

Email: <input type="text" name="email" id="email" value="" />

Comments: <textarea name="comments" id="comments" cols="25" rows="3"></textarea>

<input type="submit" value="submit" />

</form>

If you enter this into an HTML document and then open that document in a browser, the code is rendered as shown in Figure 3.4.5.

Figure 3.4.5: The first, basic form example.

Try it for yourself—enter the above code into your own sample HTML document and load it in a browser. Try playing around with the different form controls to see what you can do with them.

As you read the code, you'll see an opening <form> tag, a </form> closing tag, and some bits in between the two. The element contains two text inputs in which the page's reader can enter their name and email address, and a textarea that can be filled with a comment to submit to the site owner.

What have we got here?

<form></form>: These two tags are essential to start and end a form—without them you don't have a web form. Every form must start and end with <form> and </form> tags.

The <form> tag can have a few attributes, which will be explained in Step Two, but please do note that you can't nest forms inside each other.

<input> (should be <input /> if you're using an XHTML doc type): This tag defines the area where you can insert information. In our case above, input tags define text boxes where site visitors can type in their name and email.

Every input tag must have a type attribute to define what it will receive: the possible attribute values are text, button, checkbox, file, hidden, image, password, radio, reset or submit.

Every <input> tag must also have a name (except in special cases where the value attribute is always set to the same value as the type attribute, eg type="submit" or "reset"), which you the coder can decide on. The name attribute informs the thing the data is sent to when the form is submitted (be it a database, or an email sent to the site's administrator via a server-side script) what the name of the information in the input box is. When the form is submitted, most scripts use the name attribute to place the form data into a database or into an email that can be read by a person.

Thus, if the <input> element is for the site visitor to enter their name into, then the name attribute would be name="name" or name = "first name", etc. If the <input> tag is for an email address, then the name attribute would be name="email". To make it easier on yourself, and the person who will use the form, it is recommended that you name the <input> element in a semantic fashion.

By semantically, I mean naming it according to what its function is, as detailed above. If the input is to receive an email address, then name it name="email". If it is to be the street address of the site visitor, then name it name="street-address". The more accurate the word usage the easier it is not only for you to code the form and then perform maintenance tasks on later, but also for the person or database receiving the form to understand it. Think lean and mean with accurate meaning.

Every <input> tag should also have a value attribute. The value can be set to blank—value=""—which will tell the processing script to just insert whatever the site visitor types into the box. In the case of a checkbox, radio button, hidden, submit, or other type attributes you can set the value to equal what you want the default to be. In other cases, such as submit or hidden, you set the value to equal the final input. Examples: value="yes" for yes, value="submit" for a submit button, value="reset" for a reset button, value="http://www.opera.com" for a hidden redirect, etc.

Examples of how to use the value attribute:

A blank value attribute, which the user input determines the value of:

(i) The code says: <input type="text" name="first-name" id="first-name" value="" />

(ii) The user inputs: Jenifer

(iii) The value of first-name is sent as “Jenifer” when the form is submitted.

A predetermined value:

(a) The code says: <input type="checkbox" name="mailing-list" id="mailing-list" value="no" />

(b) The user checks the box as they wish to join the website’s mailing list.

(c) The value of mailing-list is sent as “yes” when the form is submitted.

After the two <input> elements, you can see something a bit different—the textarea element.

The folks at text area bring you a nice, new, improved space to input text into. Not your ordinary, plain old one line text box that our friend <input> provides, the text area element provides multiple lines of input, and you can even define how many lines are available to enter text into. Note the cols and rows attributes—these are required for every text area element, and specify how many columns and rows big to make the text area. The values are measured in characters.

Last but not least, you have a special <input> element with the attribute value="submit". Instead of rendering a one line text box for input, the submit input will render a submit button that, when clicked, submits the form to whichever target the form has specified to send its data to (currently this isn’t defined at all, so submitting the form will do nothing.)

Step two: Adding structure and behaviour

So, you clicked on the form #1 link above, filled it out and clicked Submit—why didn’t it do anything, and why does it look so bad and all in one line? The answer is that you haven’t structured it yet, or defined a place for the data the form is collecting to be submitted to.
Let’s go back to the drawing board, with a new form:

<form id="contact-form" action="script.php" method="post">

```
    <input type="hidden" name="redirect" value="http://www.opera.com" />
    <ul>
      <li>
        <label for="name">Name:</label>
        <input type="text" name="name" id="name" value="" />
      </li>
      <li>
        <label for="email">Email:</label>
        <input type="text" name="email" id="email" value="" />
      </li>
      <li>
        <label for="comments">Comments:</label>
        <textarea name="comments" id="comments" cols="25" rows="3"></textarea>
      </li>
      <li>
        <input type="submit" value="submit" />
        <input type="reset" value="reset" />
      </li>
    </ul>
</form>
```

This form looks like Figure 3.4.6 when rendered in a browser:

Fig. 3.4.6: The second form example—looking better, but still not perfect.

Here I have made a few additions to the basic, simple form. Let's break it down so you know what I did:

There are some new attributes inside the <form> tag. I added an id attribute to not only semantically name what this form is called, but also to provide a unique ID to identify the form so it can be more easily styled using CSS or manipulated using JavaScript if required. You can only have one of each id per page; in this case I called it contact-form.

Lights, camera, action! When you pressed the submit button in the first form and it did not do anything, this was because it had no action or method. The method attribute specifies how the data is sent to the script that will process it. The two most common methods are "GET" & "POST". The "GET" method will send the data in the page's URL (you will sometimes see URLs along the lines of these are bits of data being

transported using the “GET” method). Unless you have a specific reason to use “GET”, it is probably best to not use it if you are trying to send secure information as anyone can see the information as it is transmitted via the URL. “POST” sends the data via the script that powers the form, either to an email that is sent to the site’s administrator, or a database to be stored and accessed later, rather than in the “GET” URL.

If you are very concerned about the security of the data in the form, for example if you are submitting a credit card number to a shopping site, then you should use the https protocol with a secure socket layer (SSL). Basically, this means that data will be sent over the https protocol, not the http protocol. Have a look at the URLs next time you are paying for something on a shopping site, or using online banking—you’ll probably see https:// in your address bar, not http://. The difference is that an https connection is a bit slower to transmit than http, but the data is encrypted, so anyone hacking into the data connection can’t make any sense out of it while it is in transit. Talk to your web host for information on how they can provide you with https and SSL.

The action attribute specifies what script file the form data should be sent to for processing. Many web hosts will have a generic send mail script or other form scripts available for use (see your host’s documentation for information) that they have customized to their servers. On the other hand, you could use a server-side script that you or someone else has created to power your form. Most of the time, folks use languages such as PHP, Perl or Ruby to create a script that will process the form—you could for example send an email containing the form information, or input the form information into a database to be stored for later use.

It is outside of the scope of this course to write up a server-side script for you, or teach you how to write server-side code yourself—please inquire with your host to find out what they offer, or find a nice programmer to befriend.

Here are a few resources to get you started if you would like to investigate server-side scripting:

i. Perl: http://www.perl.com/
ii. PHP: http://www.php.net
iii. PHP documentation on Forms: http://uk3.php.net/manual/en/tutorial.forms.php
iv. Python: http://python.org/
v. Ruby: http://www.ruby-lang.org
vi. Sendmail: http://www.sendmail.org/
vii. ASP.NET: http://www.asp.net/

The second line that’s been added to our Step Two form is the “hidden” input field—this is a redirect. What?

Under the goal of separating mark-up structure from presentation and behaviour, it is ideal to use the script that will power the form to also redirect the user once the form is submitted. You don’t want your users to be left sitting there looking at the form page,

wondering what the heck to do next after they've submitted the form; I'm sure you'll agree that it is a much better user experience to instead redirect your users to a thank you page featuring "what to do next" links, after a successful form submission. This line in particular specifies that after this form is submitted, the user will be redirected to the Opera homepage.

To improve the look of the form, I have put all the form elements into an unordered list so that I can use the mark-up to line them up cleanly and then use CSS to polish the look.

Some folk would argue that you should not use an unordered list to markup a form, but use a definition list instead. Others would argue that one should not use a list at all but use CSS to style the <label> and <input> tags. I will let you research this debate and make up your own mind on which is more semantically correct. For our simple exercise I will use the unordered list.

Last but not least in step two, I've labelled the form elements. Both in terms of meaning and making the form accessible to a wide range of internet enabled devices, it is best to give all the form elements labels—check out the contents of the label elements - these labels are tied to their respective form elements by giving the input and textarea elements ids that have the same value as the labels' for attributes. This is great because it not only gives a visual indicator of the purpose of each form field on the screen, but it also gives the form fields more meaning semantically. For example, a visually impaired person using the page with a screen reader can now see which label goes with which form element. The ids can also be used to style individual form fields using CSS.

By now you are probably wondering why id attributes are included as identifiers in form elements as well as name attributes. The answer is that input elements without name attributes are not submitted to the server, so those are definitely needed. id attributes are needed to associate form elements with their corresponding label elements. You should therefore use both.

The 2nd form displays a bit better, but it has been beaten with the default ugly stick. Time to add a few more bits and bobs before applying some style.

Step three: Adding semantics, style and a bit more structure

Now I'll finish off what I started at the beginning of the article, with the following final version of my example form:

```
<form id="contact-form" action="script.php" method="post">
  <fieldset>
    <legend>Contact Us:</legend>
    <ul>
      <li>
        <label for="name">Name:</label>
        <input type="text" name="name" id="name" value="" />
      </li>
```

```
      <li>
        <label for="email">Email:</label>
        <input type="text" name="email" id="email" value="" />
      </li>
      <li>
        <label for="comments">Comments:</label>
        <textarea name="comments" id="comments" cols="25"
rows="3"></textarea>
      </li>
      <li>
        <label for="mailing-list">Would you like to sign up for our mailing list?</label>
        <input type="checkbox" checked="checked" id="mailing-list" value="Yes,
sign me up!" />
      </li>
      <li>
        <input type="submit" value="submit" />
        <input type="reset" value="reset" />
      </li>
    </ul>
  </fieldset>
</form>
```

When rendered in a browser, this form looks as shown in Figure 3.4.7.

Fig. 3.4.7: The final form example in all its glory.

The last two major elements I have added to this form are fieldset and legend. Both of these elements are not mandatory, but are very useful for more complex forms and for presentation.

The fieldset element allows you to organize the form into semantic modules. In a more complex form, you could for example use different fieldsets to contain address information, billing information, customer preference information, and so on. The legend element allows you to name each field set section.

I've also applied a little bit of CSS to this form, to style the structural mark-up. This is applied to the third form example using an external style sheet. The two most important tasks I wanted the basic CSS to do is add margins to line up the labels and input

boxes, and get rid of the unordered list's bullet points. Here is the CSS that resides in the external style sheet:

i. #contact-form field set {width:40%;}

ii. #contact-form li {margin:10px; list-style: none;}

iii. #contact-form input {margin-left:45px; text-align: left;}

iv. #contact-form textarea {margin-left:10px; text-align: left;}

What does it do? The first line styles the fieldset border to not take up the whole page; you could also set the border to none using {border: none;} if you didn't want one. The second line puts a margin of 10 pixels on the li elements to help give a little visual room between each list item. The third and fourth lines set a left margin on the input and textarea elements so that they don't crowd the labels and line up properly.

Self-Assessment Question(s)

1. Discuss the "form" in HTML
2. Convert the text below into an html document using appropriate html elements alone.

Self-Assessment Answer

1. A form online is any area where you can input information into a web page, for example entering text or numbers into a text box, checking a tick box, causing a radio button to "fill in", or selecting an option from a list. The form is then submitted to the web site when you push the submit button.

   Forms are used for the web for entering user names and passwords on login screens, commenting on blogs, filling your profile in on a social networking site, or specifying billing information on a shopping site

4.0 Conclusion

So far, you have learnt the basics of HTML but it does not end here, there is so much you can do with Hyper Text Markup that has not been covered here, and there will always be changes and updates to the syntax of HTML. So try to keep your selves up to date.

5.0 Summary

In this Unit you have learnt the following aspects have been discussed:

a. formatting text on the web page using HTML,
b. creating links on the web page,
c. HTML menu tools,
d. adding image and other page elements like font, colour, etc,

e. Background editing features in HTML.

6.0 Tutor Marked Assignments

1. Discuss the word “emphasis” in HTML.
2. List the three steps involved in creating elegant and accessible form structure with HTML.
3. Discuss background editing features in HTML.

7.0 References/Further Reading

1. Web Design Manual (2010)
2. HTML Utopia by Dan Shafer and Rachel Andrew (2006)
3. The basics of HTML http://dev.opera.com/articles/view/12-the-basics-of-html/
4. HTML Tutorial http://www.w3schools.com/html/default.asp

Unit 2

Cascading Style Sheet And Server Side Includes

Content

1.0 Introduction

Previously we learnt about how to design a full webpage but whenever we look at commercial websites like ESPN, Disney or even CNN's web pages you see a variety of typefaces, type treatments and even unique line spacing and decorations. We can accomplish these features using Cascade Style Sheet.

Cascading Style sheets are really a completely different approach to page styling and layout.

2.0 Learning Outcomes

In this Unit you will learn the following:

i. A brief History of CSS

ii. You will learn the basic rules governing CSS

iii. You will learn about and how to use the different types of CSS

iv. You will learn about Classes and IDs

v. You will learn about comments and how to modify the different ways to modify text.

vi. You will learn about Layouts and finally how to create a full page using what you have learnt with both HTML and CSS.

3.0 Learning Outcome

3.1 Cascading Style sheets (CSS)

CSS (Cascading Style Sheets) was a technology recommended by the World Wide Web Consortium (W3C) in 1996. To understand the purpose of CSS is to view it as an addition to HTML that helps simplify and improve Web page design.

In fact, some CSS effects are not possible via HTML alone. Another advantage of CSS is that it allows you to specify a style once, but the browser can apply that style many times in a document. For example, if you want some of the pictures displayed in your Web site to have a thin, blue frame around them, you can define this frame as a style in your CSS. Then, instead of having to repeat an HTML definition of the thin and blue frame — each and every time you want that particular frame — you can merely insert the CSS style as an attribute for each graphic element that you want framed.

3.1.1 Basic CSS Rules

CSS can be used to define general rules about how the elements on your Web pages behave and how they look such as size colour, opacity and so on.

Basically CSS is made up of selectors, properties and values. The format or rule is as below

Selector {property: value ;}

The selector is what is used to identify the tag. It could be a Class or ID. For example it could be h1 or (Header 1 tag), p or (paragraph tag).

The Property identifies what features you want the class or ID to have. For example colour, font face, and font-size.

The value of such property, for example, yellow for colour, Courier New for the Font-face property or even 12 for the font-size.

A good example is as below

```
<style>
H1 { font-size:16 color:blue;}
</style>
```

With this CSS rules in effect, any HTML code containing an H1 element is automatically rendered in 16-point type and coloured blue.

An example of internal CSS rules is as below

```
<html>
<head>
<style>
h1 { font-size:16pt color:blue;}
</style>
</head>
<body>
<h1>this headline is blue and 16 pt.</h1>
</body>
</html>
```

Notice the Style element <style></style> because without it, you cannot declare any CSS rules. All CSS declarations can be done between the elements.

One more thing, all CSS rules terminate with a semi column.

3.1.2 Types of CSS

Cascading Style Sheets uses a completely different approach to page styling and layout.

You need to look at where you add CSS information to your page. CSS information can be specified in three different places:

Within the specific tags in the document body

At the top of the document within a <style> block, or combined with named or containers in the document body

In one or more separate files shared across many Web pages. These may all be combined with a well-defined inheritance (which is why they are called *cascading* style sheets).

Inline CSS

CSS styles can be specified with the style attribute within almost any HTML markup tag.

For example, you can have a bold tag that also changes the colour of the text within by using

the following HTML:

this is bold and blue and this isn’t.

More commonly, styles are used within one of two otherwise empty tags called or . These two tags were introduced into HTML specifically for use with Cascading Style Sheets, so if you see them on a Web page, they’re often used like this:

<span style=”color: green”>this is green and this ain’t.

The difference between the two is that is used as a block container (it’s basically identical to and) whereas is used within a block.

Create a new html document and add this inside it

this is green and this ain’t.

this is blue and this is red.

this is yellow and this ain’t.

Save it. It should look like this when you open it on a browser.

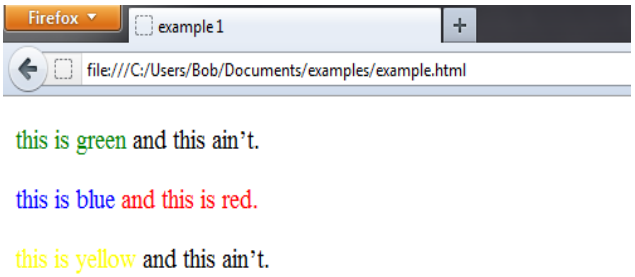


Fig. 3.1.1

The preceding example is not going to convince you that this approach is superior to the <font> tag discussed in HTML introduction. But read on, and perhaps you will begin to understand the value of style sheets.

The second way to work with CSS will doubtless catch your attention.

To a great extent, CSS enables you to *redefine how any HTML markup is rendered*. CSS also essentially allows you to create your own tags to transform a Web page into exactly the format and layout you seek.

This is accomplished by moving the style specifications into a <style> block. At the top of a page, the <style> tag might look like this:

<style type=”text/css”>

strong { color: blue }

</style>

This style specifies that throughout the subsequent document body, all occurrences of the strong element should also have a type colour change to blue. Are you starting to see where this can make your life considerably easier?

More interestingly, you can specify style *classes*, which are just like sub-tags. Imagine that I have a Web site that talks about my digital camera, a Nikon. I'm going to write about how it compares to other digital cameras, including those from Canon, Sony, and Kodak. To make the page consistent, each time that I mention a manufacturer I want to use the identical format.

Here's how I can do that with CSS:

While a variety of companies manufacture digital cameras, notably including

Kodak,

Olympus and

Sony,

there are only two companies that offer true

digital single lens reflex (SLR) cameras:

Nikon and

Casnon.

This gets interesting when you go into the <style> block and specify exactly how to format the manufacturer names.

Because these are all specified by class, the CSS notation is to preface the class name with a dot in the style block. I have highlighted it so you can see it quickly:

<style type="text/css">

.manuf { font-size: 125%; color: green }

</style>

So let's try it. Open a new notepad and insert these codes.

<head>

<title>Camera types</title>

<style type="text/css">

.manuf { font-size: 125%; color: green }

</style>

</head>

<body>

```
<h1>Camera types</h1>
<p>There are many notable camera companies these include:
<span class=”manuf”> Kodak</span>,
<span class=”manuf”> Olympus</span> and
<span class=”manuf”> Sony</span>, amongst others.</p>
<p>There are only two companies that offer true
digital single lens reflex (SLR) cameras:
<span class=”manuf”> Nikon</span> and
<span class=”manuf”> Canon</span>
</body>
```

Save the page, and open it on your browser. It should look like this

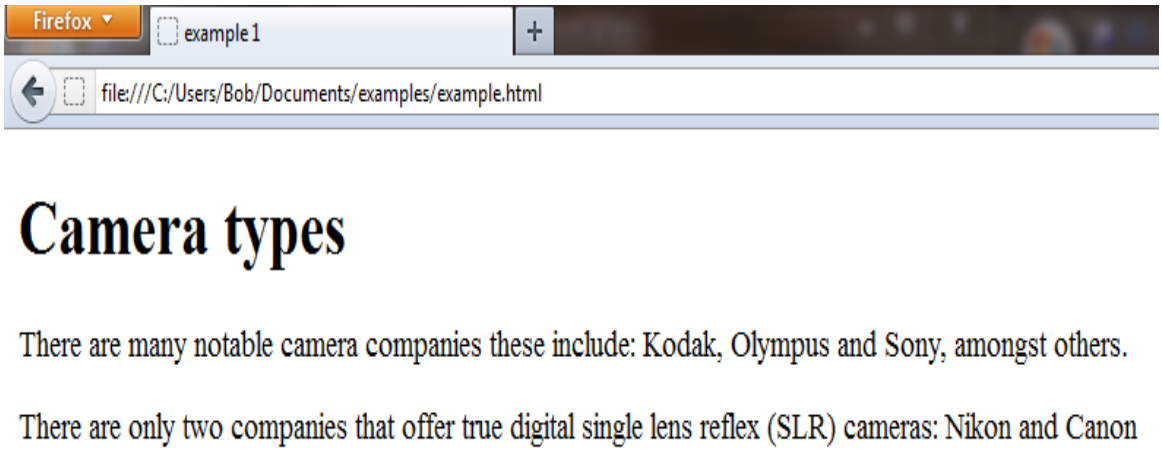


Fig. 3.1.2: Camera Type

External Style Sheet

The third way that you can work with CSS is to create a completely separate document on your Web server that includes all the styles you want to use. You can then reference that document within all the Web pages on your site. You may think that having a single style definition at the top of your page makes it easy to manage the layout of that page. Imagine how handy it would be to have a site with dozens (or hundreds!) of pages of material, all using the appropriate div and span tags and classes. Add to that the capability to change the style across all occurrences of a class, on all pages, with a single edit!

To reference a separate style sheet, use the link tag:

```
<link type=”text/css” href=”mystyles.css” />
```

This refers to a separate style sheet called mystyles.css, stored in the same directory on the same server as the page that contains this link reference. You can also use a

fully qualified URL. This is how I reference one of my CSS style sheets on my server: <link type="text/css" href="http://www.intuitive.com/library/shared.css" />

The .css file should contain everything you'd otherwise put between the Style element <style></style>.

Class and ID

Classes

Classes help to sub-divide a tag or style into several alternatives. Similar to divide fruits into sub categories like Mangoes, Oranges, Bananas and so on.

Remember this example?

.manuf { font-size: 125%; color: green; }

We use the **.manuf** to set the class name and it's properties and values are in the curly bracket **{ font-size: 125%; color: green; }**

But that's not all, whenever we call the class in our html page we will automatically be giving the properties of that class name to whoever calls it

 Kodak,
 Olympus and
 Sony, amongst others.
There are only two companies that offer true
digital single lens reflex (SLR) cameras:
 Nikon and
 Canon.

Let's use what we have learnt to create another page and add these codes.

As always, I have taken the time to highlight the classes.

<html>
<head>
<title>You need class</title>
<style type="text/css">
.clas { color: green; font-face: arial;}
.alert{ color: red; font-face: lucida sans;}
.mumu{ color:orange; font-face: Courier New;}
<title>You need class</title>
<style type= "text/css">
</style>
</head>
<body>
<h1>It's all about class</h1>
<**div class="clas"**>Lorem ipsum dolor sit amet,
<**div class="mumu"**> consectetuer adipiscing elit.
Praesent aliquam, justo
convallis luctus rutrum,

erat nulla fermentum diam,
at nonummy quam ante ac quam.
Maecenas urna purus, fermentum
id, molestie in, commodo porttitor,
felis.

<**div class= "alert"**>Nam blandit quam ut lacus.
</body>
</html>

Once you have the saved the page and opened it on your browser, it should look like this

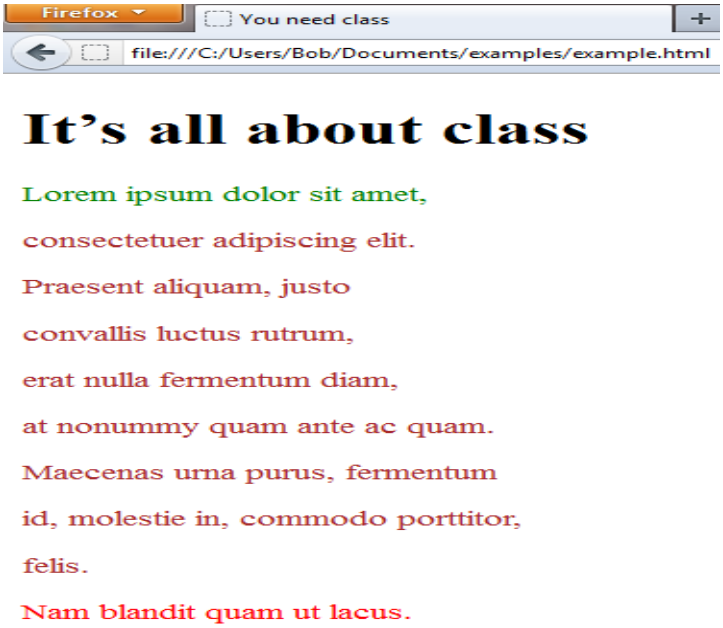


Fig. 3.1.3: Class and ID

ID

IDs are similar too classes, too, are independent of specific document elements. The only difference between Ids and Class is that Ids are only supposed to work only *once* on a given Web page.

But truly, ID selectors don't work as advertised. In *practice, the web* browsers just ignore this "use it only once" rule

While class may use **.maunf { property: value; }**

ID uses **#manuf {property: value; }**

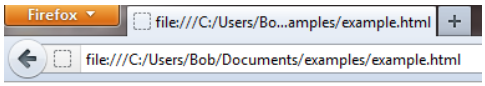
The difference between class and ID is that class uses **dot** while ID uses **#.**

Once again open a new notepad and add these codes:

<html>
<head>
<style>

```
#highlight {color: yellow; font-style: italic;}
</style>
</head>
<body>
<p>ordinary default text </p>
<p id="highlight">highlighted text </p>
<p id="highlight">second attempt to use this id fails
</p>
<h1 id="highlight">highlighted text </h1>
</body>
</html>
```

Save this page and open it on your browser



ordinary default text

highlighted text

second attempt to use this id fails

highlighted text

A brief word on CSS comments

One thing to know early on is how to comment in CSS. You add comments by enclosing them in /* and */. Comments can span several lines, and the browser will ignore these lines:

```
<style>
/* These are basic element selectors */
h1{font-size:16pt color:blue;}
</style>
```

Text Formatting using CSS

Colour

You can set the color of text with the following:

```
{Color: value;}
```

You can use the following as values for color:

Colour name: Red, Black, Blue

Hexadecimal number: #ff0000, #000000

RGB color code:(rgb(255, 0, 0), rgb(0, 0, 0))

Letter Spacing

You can adjust the space between letters in the following manner. Setting the value to 0, prevents the text from justifying. You can use negative values.

Letter-spacing: value;

Possible values are

normal

length

Examples

.example{letter-spacing:5px;}

#example{letter-spacing:normal;}

Each text in this ID (#example) will be spaced normally.

Text Align

You can align text with the following:

{text-align: value;}

Possible values are

{text-align:left;}

{text-align:right;}

{text-align:center;}

{text-center:justify;}

Text Decoration

You can decorate text with the following:

{text-decoration: value;}

Possible values are

none

underline

overline

line through

blink

Examples:

p{text-decoration:none;}

strong{text-decoration:underline;}

#ace{text-decoration:blink;}

.para{text-decoration:line through;}

Text Indent

You can indent the first line of text in an HTML element with the following:

{text-indent: value;}

Possible values are

length

percentage

Examples:

{text-indent:12;}

{text-indent:12%;}

This text is indented 10px pixels.

Text Transform

You can control the size of letters in an HTML element with the following:

text-transform: value;

Possible values are

none

capitalize

lowercase

uppercase

Examples:

{text-transformation:none;}

{text-transformation:capitalize;}

{text-transformation:lowercase;}

{text-transformation:lowercase;}

White Space

You can control the whitespace in an HTML element with the following:

{white-space: value;}

Possible values are

normal

pre

nowrap

{white-space:none;}

{white-space:pre;}

{white-space:nowrap;}

Word Spacing

You can adjust the space between words in the following manner. You can use negative values.

{word-spacing: value;}

Possible values are

normal

length

Example:

h1{font-size:5pt color:blue;}

So let’s put all of what we have learnt about CSS text editting together i.e in a code.

Open two new notepads

One is for you css (external rules) and the other is for your HTML codes

In the CSS page add these codes

h1, h2, h3{ text-align:center;

color:teal; font-family:Arial;

white-space:pre; letter-spacing:3px;

text-transform:capitalize; text-indent:12px;

text-decoration:underline;}

p{text-align:center; color:#33F;

font-family:Gadget; text-decoration:none;

font-size:12;}

h6{text-align:center; color:#33C;

font-family:Lucida Sans Unicode;}

#onlyone{ text-decoration:line-through;}

.warning{ text-align:center; color:red;

letter-spacing:8px; text-decoration:blink; text-indent:4;}

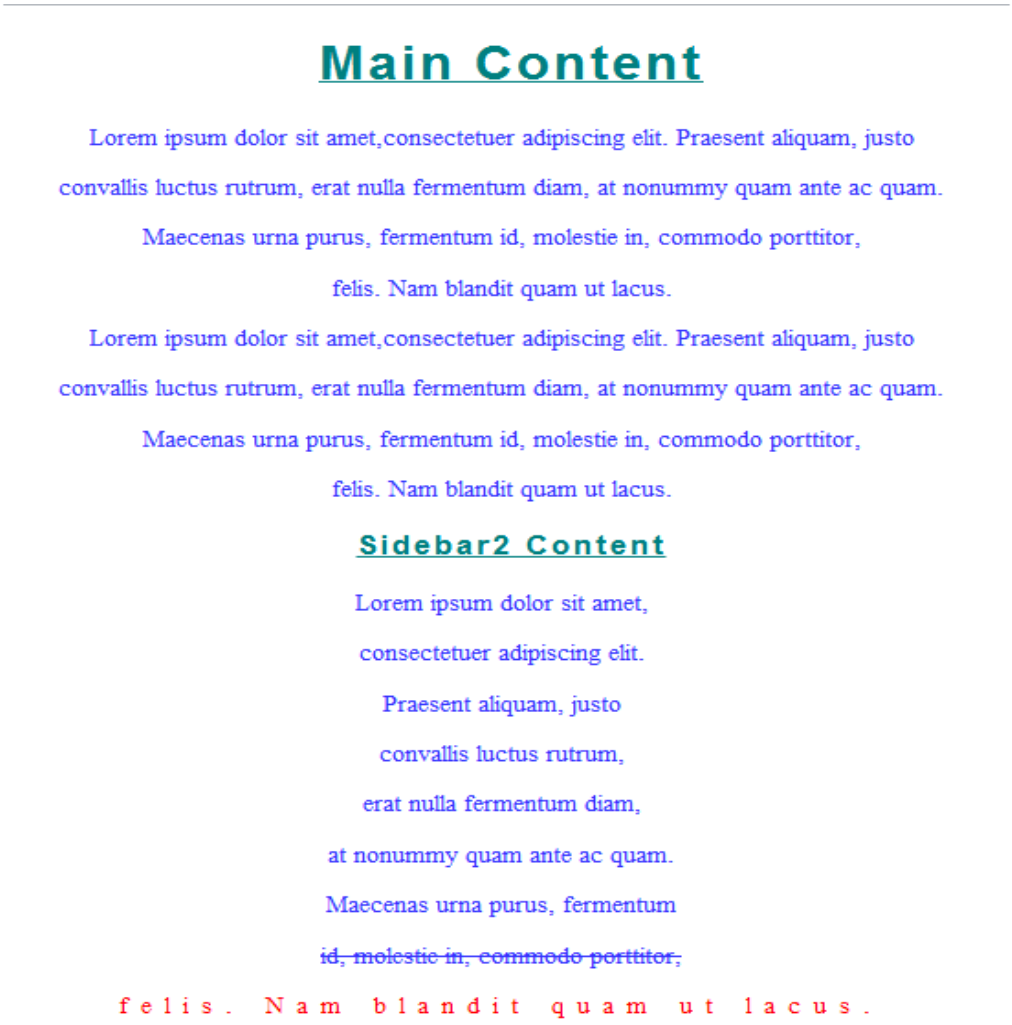
Remember to save the notepad as **.CSS**

Now Open the second page and insert these codes:

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

```
<title>CSS TEXT FORMAT</title>
<link href="newstyle.css" rel="stylesheet" type="text/css" />
</head>
<body>
<h1>Main Content</h1>
<p>Lorem ipsum dolor sit amet,consectetuer adipiscing elit. Praesent aliquam, justo</p>
<p>convallis luctus rutrum, erat nulla fermentum diam, at nonummy quam ante ac quam.</p>
<p>Maecenas urna purus, fermentum id, molestie in, commodo porttitor,</p>
<p>felis. Nam blandit quam ut lacus.</p>
<p>Lorem ipsum dolor sit amet,consectetuer adipiscing elit. Praesent aliquam, justo</p>
<p>convallis luctus rutrum, erat nulla fermentum diam, at nonummy quam ante ac quam.</p>
<p>Maecenas urna purus, fermentum id, molestie in, commodo porttitor,</p>
<p>felis. Nam blandit quam ut lacus.</p></td>
<td valign="top" width="200" bgcolor="#FFFFFF" >
<h3>Sidebar2 Content</h3>
<p>Lorem ipsum dolor sit amet,</p>
<p>consectetuer adipiscing elit.</p>
<p>Praesent aliquam, justo</p>
<p>convallis luctus rutrum,</p>
<p>erat nulla fermentum diam,</p>
<p>at nonummy quam ante ac quam.</p>
<p>Maecenas urna purus, fermentum</p>
<p id="onlyone">id, molestie in, commodo porttitor,</p>
<p class="warning">felis. Nam blandit quam ut lacus.</p>
</body>
</html>
```

Now save the page as a .html and open it on your browser. It should look like this



3.1.3 Creating Layouts Using CSS

Unlike HTML that depends on tables to design layout, there are several features that can be used to develop the perfect layout for your page.

Divsions

Divisions are a block level HTML element used to define sections of a HTML file. A division can contain all the parts that make up your website. Including additional divisions, spans, images, text and so on.

You define a division within an HTML file by placing the following between the <body></body> tags:

Site contents go here

Though most likely you will want to add some style to it. You can do that in the following fashion:

Site contents go here

The CSS file contains this:

```
#container{
  width: 70%;
  margin: auto;
  padding: 20px;
  border: 1px solid #666;
  background: #ffffff;
}
```

Now everything within that division will be styled by the “container” style rule, I defined within my CSS file. A division creates a linebreak by default. You can use both classes andIDs with a division tag to style sections of your website.

BORDERS

A border is a frame, just a line usually, that surrounds an element. However, you can selectively leave out any of the four lines of a border or define them each differently using the border-top, border-right, border-bottom, and border-left properties individually.

A border can be used to surround the content that you’ve specified for an element. Any optional margin specified is not surrounded by the border, but instead separates this element from surrounding elements.

Borders can be given values indicating how you want them to look i.e. colour, thickness, and style. The thickness or width of the border can be specified in the usual CSS variety of ways, including the default medium, which are two or three pixels wide. The default colour of a border is the text colour of the element, or if an element has no text, the text colour of the element’s parent is inherited.

A simple border can be specified with this format:

{border-left:value;}

{border-right:value;}

{border-top:value;}

{border-bottom:value;}

Another simple way to specify all the borders is to specify them all at the same time in a clock wise direction.

{border left-value top-value right-value bottom-value;}

To put this into a better perspective, it will look like this.

{border: 250 750 250 750;}

There are other things you can do with the border such as color (border-color) and change the thickness of the frame.

MARGIN

Margin separates the paragraph element from a nearby element; Just as in your Word Processor, Margin organises your text within or outside the borders.

Margins share a similar format to borders.

{margin-left: value;}

{margin-right:value;}

{margin-top:value;}

{margin-bottom:value}

Examples

{margin-left:10;}

{margin-right:10;}

{margin-top:10;}

{margin-bottom:10;}

Another simple way to specify all the borders is to specify them all at the same time in a clock wise direction.

{Margin: left-value top-value right-value bottom-value;}

To put this into a better perspective, it will look like this.

{margin: 250 750 250 750;}

PADDING

Now to add padding just like margin and border the format is thesame.

{Padding-left: value;}

{Padding-right:value;}

{Padding-top:value;}

{Padding-bottom:value}

Examples

{Padding-left:10;}

{Padding-right:10;}

{Padding-top:10;}

{Paddng-bottom:10;}

And finally you can use all four values together with one property i.e. {Padding: 2 1 2 1;}

Positions

The trick when using CSS positioning is to create classes or IDs that specify where you want your zones located.

Absolute Position

Absolute Position lets you determine an element's position and location by specifying the Left, Right, Top and Bottom in Pixels (PX), Centimetres (cm) and Percentage (%). It can be used to specify the backgrounds.

Relative Position

A relative Positioned element appears relative to it's current position in HTML4. But Unlike Absolute Positioning, other page elements accommodate the old html placement of a relatively positioned element.

Fixed Position

A fixed position is locked in place on the screen.
Now let's build our layout with what we have learnt so far
Open a new page for your CSS external rules and insert these:

```
.wrapper{margin: 20px 30px 20px 30px; background-color:#CF9; padding: 1px 2px 1px 2px; width:900px;}
.wrapper2{ margin:40px 30px 40px 30px;}
h1, h2, h3{ text-align:center;
color:teal; font-family:Arial;
white-space:pre; letter-spacing:3px;
text-transform:capitalize; text-indent:12px;
text-decoration:underline;}
p{ text-align:justify; color:#33F;
font-family:Gadget; text-decoration:none;
font-size:12; padding-left:6px; padding-right:6px; }
h6{text-align:center; color:#33C;
font-family:Lucida Sans Unicode;}
#onlyone{ text-decoration:line-through;}
.warning{ text-align:center; color:red;
letter-spacing:8px; text-decoration:blink; text-indent:4;}
```

Save this as newstyle.css
Open a new notepad and insert these HTML codes

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>CSS TEXT FORMAT</title>
<link href="newstyle.css" rel="stylesheet" type="text/css" />
```

```
</head>
<body bgcolor="#CCCCCC">
<div align="center" class="wrapper">
<hr/>
<div align="right" class="wrapper2"><a href="example.html">Home</a> <a href="example2.html"> My Holiday Pictures</a>
</div>
<hr />
<h1>Main Content</h1>
<p>Lorem ipsum dolor sit amet,consectetuer adipiscing elit. Praesent aliquam, justo convallis luctus rutrum, erat nulla fermentum diam, at nonummy</p>

<p> quam ante ac quam. Maecenas urna purus, fermentum id, molestie in, commodo porttitor, felis. Nam blandit quam ut lacus. </p>

<p>Lorem ipsum dolor sit amet,consectetuer adipiscing elit. Praesent aliquam, justo convallis luctus rutrum, erat nulla fermentum diam, at nonummy </p>

<p>quam ante ac quam. Maecenas urna purus, fermentum id, molestie in, commodo porttitor, felis.Nam blandit quam ut lacus. fermentum diam, </p>

<p>at nonummy</p>

<p>Lorem ipsum dolor sit amet,consectetuer adipiscing elit. Praesent aliquam, justo convallis luctus rutrum, erat nulla fermentum diam, at nonummy</p>

<p> quam ante ac quam. Maecenas urna purus, fermentum id, molestie in, commodo porttitor, felis. Nam blandit quam ut lacus. fermentum diam,</p>

<p>at nonummy. Lorem ipsum dolor sit amet,consectetuer adipiscing elit. Praesent aliquam, justo convallis luctus rutrum, erat nulla fermentum diam,</p>

<p> at nonummy quam ante ac quam. Maecenas urna purus, fermentum id, molestie in, commodo porttitor,felis. Nam blandit quam ut lacus.</p>

</td>

</td>

<td valign="top" width="200" bgcolor="#FFFFFF" >

<h3>Sidebar2 Content</h3>

<p>Lorem ipsum dolor sit amet,consectetuer adipiscing elit. Praesent aliquam, justo convallis luctus rutrum, erat nulla fermentum diam, at </p>

<p>nonummy quam ante ac quam. Maecenas urna purus, fermentum id, molestie in, commodo porttitor, Lorem ipsum dolor sit amet,</p>

<p>consectetuer adipiscing elit. Praesent aliquam, justo convallis luctus rutrum, erat nulla fermentum diam, at nonummy quam ante ac quam. Maecenas </p>

<p>urna purus, fermentum id, molestie in, commodo porttitor, felis. Nam blandit quam ut lacus. </p></td>
```

felis. Nam blandit quam ut lacus.

</body>

</html>

Save these codes as an HTML Document and open it on your browser. It should look like this:

Now let's try one more.

Take a look at the image illustrating a real page called Punchonline (www.punchontheweb.com) let's try to design and modify it a bit.

First let's create a css page and insert these codes:

.wrapper{margin: 20px 30px 20px 30px; background-color:#FFF; padding: 1px 2px 1px 2px; width:1100px; height:750px;}

```
.search{ padding-top:6px; padding-right:6px; margin-top:6px; margin-right:3px;}
.sideimage{
        position:absolute;
        left: 692px;
        top: 22px;
        width: 462px;
        height: 69px;
}
#maincontent{ margin:40px 30px 40px 10px; height:inherit; width:inherit;}
.links{position:absolute;}
.sidebar{
        margin:20px 0px 20px 20px;
        background-color: #FFF;
        border:#666;
        position:absolute;
        left:766px;
        top: 295px;
        width: 348px;
        height: 420px;
}
h1, h2, h3{ text-align:left;
color: #333; font-family:Arial;
white-space:pre; letter-spacing:3px;
text-transform:capitalize; text-indent:12px;
text-decoration: underline;}
p{ text-align:justify; color:#33F;
font-family:Gadget; text-decoration:none;
font-size:12; padding-left:6px; padding-right:6px; }
h6{text-align:center; color:#33C;
font-family:Lucida Sans Unicode;}
```

Now for the HTML codes. Insert these codes in a new notepad

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

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

<head>

<title>CSS TEXT FORMAT</title>

<link href="newstyle.css" rel="stylesheet" type="text/css" />

</head>

<body bgcolor="#666666">

<div class="wrapper">

<div align="right"></div>

<div><img src="punch.jpg" /> <div class="sideimage"></div>

  <div class="sideimage"></div>

  <span class="sideimage"><img src="First Bank Diaspora Banking.jpg" /></span>

<div><a href="home">Home</a> <a href="AM Business">AM Business</a> <a href="Business">Business</a> <a href="Sport">Sport</a> <a href="Opinion">Opinion</a> <a href="Metro">Metro</a> <a href="Health">Health</a> <a href="Feature">Feature</a> </div>

<hr color="red" />

<p> you are here: <em>Home / AM Business / Mistakes entrepreneurs make</em></p>

<div align="right">

<hr color="red" />

<img src="forum.jpg" /> <img src="jobs.jpg" /></div>

<div id="maincontent">

<h2>Mistakes entrepreneurs make</h2>

<h5>AUGUST 15, 2012 BY MBA.TUCK.DARTMOUTH.EDU </h5>

<b>- Sticking with one and only one  idea for too long</b>

<p>While a single idea may be your catalyst to entering a market, don’t be afraid to continue to explore new ideas

<p> and options. Remain open-minded, and explore new ideas to see which ones will pan out into feasible market

<p> opportunities.

  <p>- Being product-driven, not customer-driven
```

In the world of capitalism, the customer is king. Even if your product is faster, better, or stronger than the

competition’s, if it isn’t what your customers want, then they won’t buy it. It’s that simple.

And to know what your customers want, ask them! Understanding what your customer

 wants and needs should be your

 number one priority.

Two suspected bombers blow selves up

<hr />

 Govt lawyers advise FG to try Lawan, Otedola

<hr />

 Presidential panel wants police ministry scrapped ...says IG earns less than D-G SSS, EFCC chair

<hr />

Bayelsa flag, Ogoni independence declaration illegal – NBA

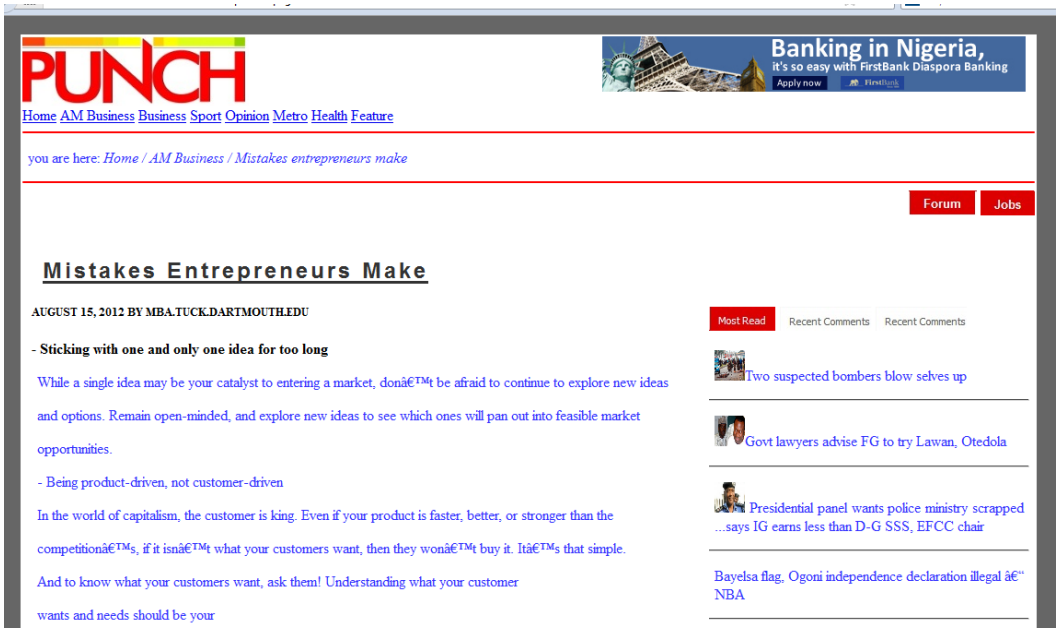
<hr />

Holyfield under “arrest”

</body>

</html>

You try to get some images to inset into it but once you save it it should look like this:



Self-Assessment Question(s)

1. In your own words explain the history of CSS
2. What's the difference between id and Class?
3. Yes or no: Will specifying a border around an element will also provide for a gutter around the content of that element, by default?
4. Why use Cascade style sheet over HTML alone?
5. What is the difference between static and dynamic web pages?

Self-Assessment Answer

1. CSS stands for Cascading Style Sheets. It was a technology recommended by the World Wide Web Consortium (W3C) in 1996. Cascade style sheet is an addition to HTML that helps simplify and improve Web page design.

2. Classes help to sub-divide a tag or a style into several alternatives Similar to dividing fruits into mangoes, oranges. While Ids also help sub – divide a tag but Ids are supposed to be used only once

3. No

4. Some CSS effects are not possible via HTML alone. Another advantage of CSS is that it allows you to specify a style once, but the browser can apply that style many times in a document.

5. A static website is one in which the content i.e. the HTML and graphics are served to the visitor or client at the same time unless the person who creates

it manually changes the copy of it. While Dynamic pages are web pages created by dynamic web technolonies used to create dynamic sections that displays different result depending on the fed values from the database.

3.2 Server side include

History of Server Side Scripting

Server-side scripting was first used in early 1995 by Fred DuFresne while developing the first web site for Boston, MA television station WCVB. The patent was issued in 1998 and is now owned by Open Invention Network (OIN). In 2010 OIN named Fred DuFresne a "Distinguished Inventor" for his work on server-side scripting.

Server-side scripting was first used in early 1995 by Fred DuFresne while developing the first web site for Boston, MA television station WCVB.

Server-side scripting is a techiquse used by web designers which involves embedding sripting in a HTML source code which results in the custom HTML being processed on the Web server before the html is sent to the client's machine over the internet.

Server-side scripting is used to provide client or users with an interface and to limit client access to proprietary datavase or other data sources.

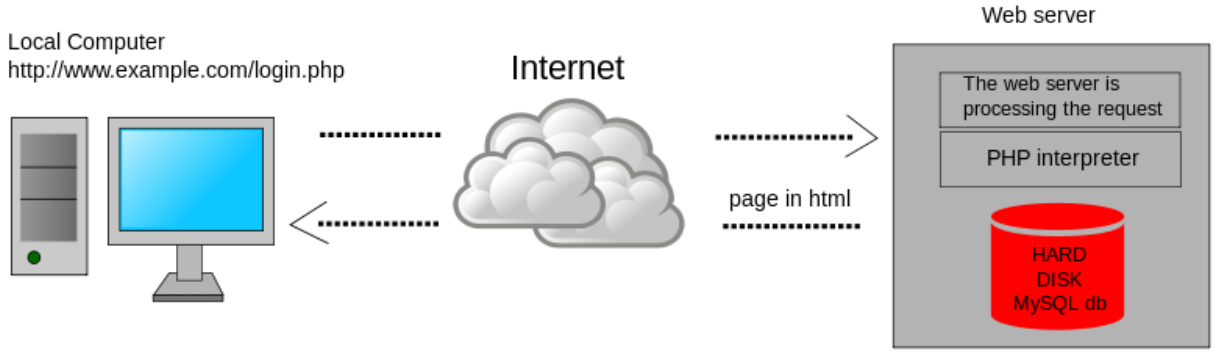


Fig. 3.2.1: Internet Component

Server-side scripting also enables the website owner to reduce user access to the source code of server-side scripts which may be proprietary and valuable in itself. The down-side to the use of server-side scripting is that the server website computer needs to provide most of the computing resources before sending a page to the client computer for display via its web browser.

Functions of Server side Scripts

1. Dynamically edit, change or add any content to a Web page
2. Respond to user queries or data submitted from HTML forms

3. Access any data or databases and return the result to a browser
4. Customize a Web page to make it more useful for individual users
5. Provide security since your server code cannot be viewed from a browser.

Server-Side Scrfipting Language

Server-Side Scrfipting Language include the following

i. ASP (*.asp, *.aspx)
ii. C (*.c, *.csp)
iii. ColdFusion Markup Language (*.cfm)
iv. Java via JavaServer Pages (*.jsp)
v. JavaScript using Server-side JavaScript (*.ssjs, *.js)
vi. Lua (*.lp *.op)
vii. Perl CGI (*.cgi, *.ipl, *.pl)
viii. PHP (*.php)
ix. Python via Django (*.py)
x. Ruby, e.g. Ruby on Rails (*.rb, *.rbw)
xi. SMX (*.smx)
xii. Lasso (*.lasso)
xiii. WebDNA (*.dna,*.tpl)
xiv. Progress WebSpeed (*.r,*.w)
xv. Note: The asterixs * stands for the name of the file.

Self-Assessment Question(s)

1. What can Server Scripts Do?
2. List at least ten (10) server side scripting languages

Self-Assessment Answer

1. Server-side scripting is a techiquse used by web designers which involves embedding sripting in a HTML source code which results in the custom HTML being processed on the Web server before the html is sent to the client's machine over the internet.
2. - Lasso (*.lasso)
   - WebDNA (*.dna,*.tpl)

- Progress WebSpeed (*.r,*.w)
- Java via JavaServer Pages (*.jsp)
- JavaScript using Server-side JavaScript (*.ssjs, *.js)
- Lua (*.lp *.op)
- Perl CGI (*.cgi, *.ipl, *.pl)
- PHP (*.php)
- Python via Django (*.py)
- Ruby, e.g. Ruby on Rails (*.rb, *.rbw)

4.0 Conclusion

So far, you have learnt the basics of CSS but it does not end here, there is so much you can do with Cascading Style Sheet that has not been covered here, and there will always be changes and updates to the syntax of CSS. So try to keep yourselves up to date.

5.0 Summary

In this Unit 1, you have learnt the following aspects discussed:

i. history of CSS
ii. basic rules governing CSS
iii. different types of CSS
iv. classes and IDs
v. comments and different ways to modify text.
vi. layouts and how to create a full page using what you have learnt with both HTML and CSS.

6.0 Tutor Marked Assignments

1. If you want to place a rule under the text of each heading in a document, which property would you use?
2. The in CSS rule {margin: 12 14 11 13 ;} what does 12 stand for?
3. From the above what do 11 stand for?
4. From the CSS rules below, what is wrong the comment statements?
5. Briefly, narrate the history of server side scripting.

7.0 References/Further Reading

HTML Utopia (2006)

CSS tutorial http://www.w3schools.com/css/default.asp

Creating Cool sites with HTML, XHTML and CSS

CSS Basics http://dev.opera.com/articles/view/27-css-basics/

CSS- the missing manual 2nd Edition David Swyer Mcfarland

Unit 3

Graphics GIF

Content

1.0 Introduction

According to WebsiteOptimization.com, On average Images make up 50% of the average web page, hence it is important to minimize the effect these images will have on the speed of the web page. If the web images are too heavy, the site will take too much time to load them and in some occations it may not even load them at all and if

the image is too light, the quality of the image will be in question hence making the web page look less sofisticated as it ought to.

Knowing when to use different image formats can make the difference in the quality and the speed of the web page. Therefore, in this section we will look at different kind of image formats and multimedia.

2.0 Learning Outcomes

At the end of this Study Unit, you should be able to:

i. Discuss graphics GIF, JPEG and PNG formats,

ii. Design graphics with palette,

iii. Add animated GIFs , multimedia and interactivity to your web page.

3.0 Learning Content

3.1 Graphics GIF

Web Image

In general, Palette-based or index-color format like GIF or PNG are best for flat t-colour arts like buttons, logo or cartoons. This is because the do not take too long to load and are generally small in size. While for Smooth toned sites, images can be saved as JPEG.

GIF Images

Use GIF files for images that have a small, fixed number of colors. GIF files are always reduced to no more than 256 unique colors. The compression algorithm for GIF files is less complex than for JPG files, but when used on flat color images and text it produces very small file sizes.

The GIF format is not suitable for photographic images or images with gradient colors. Because the GIF format has a limited number of colors, gradients and photographs will end up with banding and pixelation when saved as a GIF file.

3.2 JPEG, PNG formats

JPEG Formats

The term JPEG is an acronym for the Joint Photographic Expert Group and it is the most common image format used by digital cameras.

The JPEG standard specifies the codec, which defines how an image is compressed into a stream of bytes and decompressed back into an image, but not the file format used to contain that stream. Jpeg compression algorithm is at it's best when used on photographs and paintings or realistic scenes and smooth variations of times and colours. Jpeg is not suited for animated images, line drawings.

Jpeg is also not suited for files that will undergo multiple edits as the image quality will be lost.

The most common filename extension for JPEG is .jpeg .

Fig. 3.2.1: Example of an image saved as .Jpeg file

Examples of Softwares you can use to convert images from one image file format to another include the following:

i. Adobe Fireworks
ii. Adobe Flash Professional CS6
iii. Swish Max4
iv. Corel Draw
v. Adobe Photoshop

PNG Formats

Portable Network Graphics (PNG) was developed based on GIF, for lossless compression and for display of images on the web. Unlike GIF, PNG supports 24 bit images and produces background transparency without jagged edges; however, some older web browsers do not support PNG images. PNG format supports RGB, Indexed Color, Grayscale, and Bitmap mode images. PNG also preserves transparency in grayscale and RGB images.

i. PNG can be catoegorised into the following

ii. Index PNG

iii. Full color PNG

iv. Animated PNG

Fig. 3.2.2: Example of an Image saved as .png

3.3 Designing Graphics with Palette

3.3.1 Colors

Web designers want to add a signature touch to their sites with many design elements. Fortunately, a designer can add colour to a site without losing accessibility and usability if the site is designed with those capabilities in mind. While many designers feel comfortable designing a site for many users, those same designers might feel inadequate when it comes to choosing colours and graphics.

Colors, Tints, and Shades

Colours, or hues, have historically been divided into primary, secondary, and tertiary colours. The primary colours consist of red, yellow and blue, and they're called primary colours because you don't need to mix colours to make these three hues. If you want to translate these colours into web colours, you can recreate them using their hex (hexadecimal) equivalents of #ff0000, #ffff00, and #0033cc as shown below:

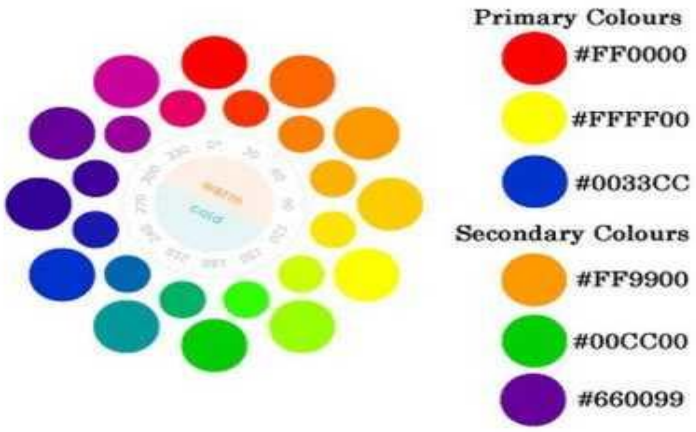


Fig. 3.3.1: The primary and secondary colours, and their hex equivalents

Secondary colours are mixed from primary colours, and those colours are as follows:

- Red + Yellow = Orange (#ff9900)
- Yellow + Blue = Green (#00cc00)
- Blue + Red = Violet (#660099)

Tertiary colours are mixed from the secondary colours, and they lie between the primary and secondary colours shown on the wheel above. Although web colours differ from regular "painters" colours, it might help to get hold of a colour wheel (as seen in the image below) to have at hand while you learn about various colour schemes. In addition, a colour wheel will show all the tints, tones and shades so you can begin to realize the colour possibilities you have at hand. Some more important terms to learn are as follows:

- Tint – The resulting colour when white is added
- Tone – The resulting colour when gray is added
- Shade – The resulting colour when black is added

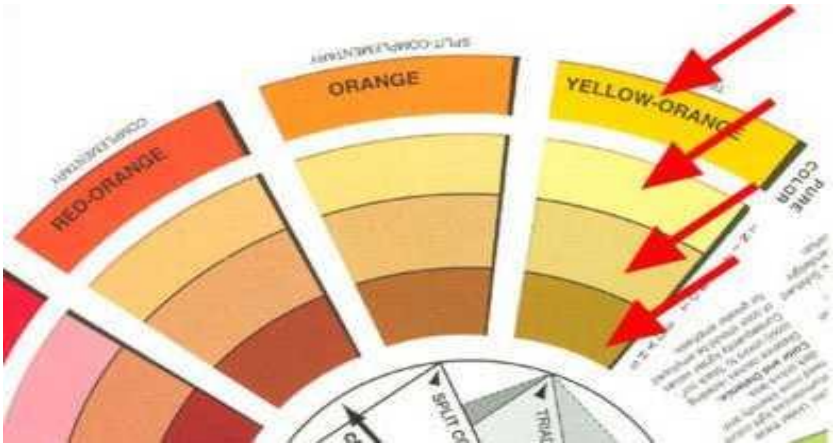


Fig. 3.3.2: A real colour wheel

The arrows indicate different things as follows:

- Outermost band – tertiary colour of yellow-orange (yellow + orange)
- Second band – the tint of that tertiary colour (white added)
- Third band – the tone of the colour (gray added)
- Innermost band – shade on the print wheel (black added)

As you can see from the colour wheel shown above, the amount of white, gray and black added to a colour are minor—just enough to alter the original colour and to create what is known as a **monochromatic colour scheme**.

3.3.2 Monochromatic Color Schemes

Colour schemes have been around for centuries, so there's no need to reinvent the colour wheel. Although web colour differs from print colour, the concepts are the same. You just exchange hex numbers for colour names, and match then as closely as possible. One online tool I suggest using to help out with this is the Colour Scheme Designer, as seen in Figure 3, which allows you to determine colour schemes quickly and easily, and even determine whether the colours you've chosen provide enough contrast for low-vision or colour-blind users.

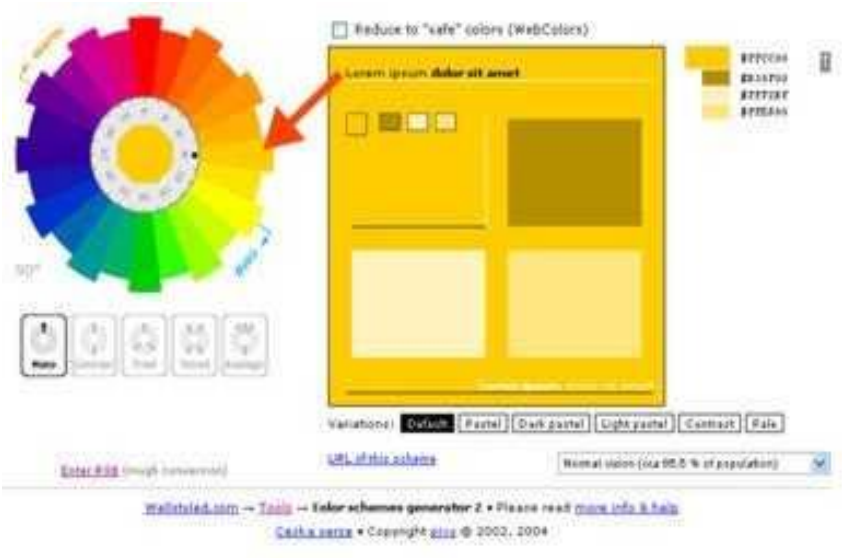


Fig. 3.3.3: Colour Scheme Generator II.

If you want more help deciding whether the colours you've chosen provide a good enough contrast, try out the Contrast Analyser from the Paciello Group. This tool checks the contrast between foreground and background colours.

To achieve the tint, tone and shade for the yellow-orange colour at the online colour generator, first select the colour that the arrow points to in the image shown above. Then, select **Mono** in the panel located under the colour wheel and **Default** in the panel in the box to the right. Also select **Normal Vision** in the selection from the pull-down menu at the bottom right. Do not check the "reduce to ‘safe’ colours" box above the colour box unless you’re a purist.

Note: The term "web-safe colours" comes from a time when monitors could display 256 colours, only 216 of which were the same across Windows/Mac/Unix platforms, hence the "web safe" monicker. While some purists still stick with the "Web-safe colour palette", modern browsers are capable of handling what is known as "24-bit" colour. Actual 24-bit colour at ten to eleven bits per channel produces 16,777,216 distinct colours. In other words, it’s safe to say that the "Web safe" colour palette rarely is needed anymore.

When talking about the monochromatic colour scheme. The results you should receive by following the steps described above are: yellow-orange (#FFCC00), tint (#FFF2BF), tone (#FFE680), and shade (#B38F00). These hex numbers are much more reliable than any guesses you will make by trying to match a tangible colour wheel to the backlit screen of a Web browser. And, as the "Mono" suggests, this scheme translates to a monochromatic colour scheme, as seen below.

Fig. 3.3.4: A monochromatic colour scheme.

A monochromatic colour scheme equates to one colour and all its tints, tones and shades. While this scheme is the easiest to use, it doesn’t provide much excitement

in a Web design for many designers. Instead, you may want to explore other schemes to add pizzaz to your links, images, and banners.

3.3.3 Complementary Colour Schemes

The next colour scheme family to explore is the complementary scheme, where you match up colours that lie directly opposite each other on the colour wheel, as seen below.

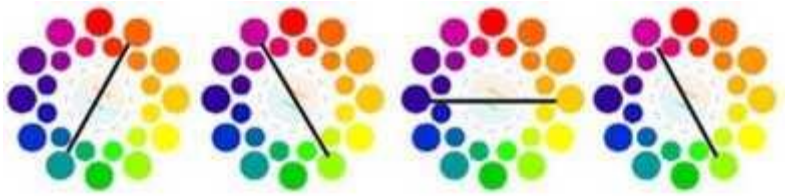


Fig. 3.3.5: Examples of complementary colour schemes.

When you choose one colour and its opposite colour, you also include all the tints, tones and shades of both colours. This provides a wider range of choices, and it translates well with the online colour tool—as seen below.

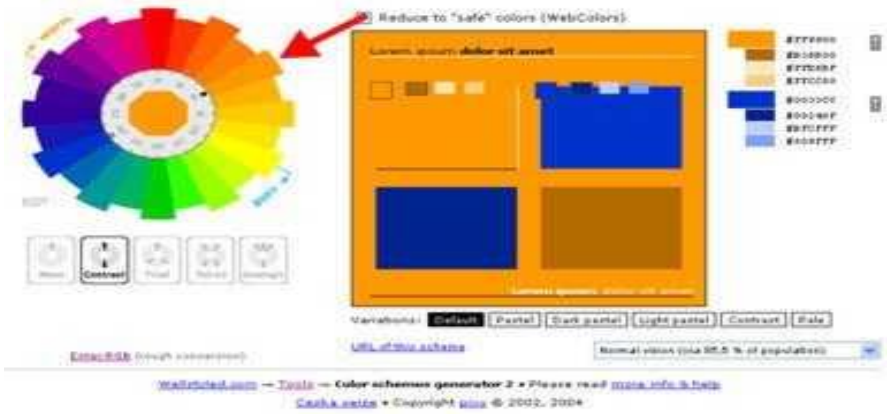


Fig. 3.3.6: A complementary colour scheme inside the online colour tool.

In the image above, I have chosen an orange colour with the opposite complementary colour of blue. The settings I chose to get to this scheme include the **Contrast** setting at lower left, the **default** on the menu below the generator, and **normal vision**. Notice that the main colour selected is marked by a black dot on the inner disc of the colour wheel (both above and online at the generator site) and that the opposite, complementary colour that was automatically picked for you is marked with a hollow circle in the inner rim. These markings make it easier for you to analyze your colour scheme.

This colour generator makes it easy for you to choose colours for links, visited links, and even images as it provides the hex colours for you at the upper right. You can mix and match any pure colour (the colour at the top) and its tint, tone or shade and feel great about choosing a solid colour scheme.

Fig. 3.3.7: The Greenpeace site - a good example of a complementary colour scheme.

Greenpeace USA is one of many sites that use a contrasting colour scheme. Yes, you see yellows and oranges, but the predominant colours are green and red - two colours that are directly opposite each other on the colour wheel. You almost can't go wrong with this complementary colour method. In fact, the use of a "warm" and "cool" colour combination makes a site zing with colour contrast.

3.3.4 Warm vs. Cool Colours

Complementary colour schemes are great to use in web sites, as they also contain both warm and cool colours. The use of these colours provides contrast, and it's easy to remember which colours are "warm" and which colours are "cool" as you can see in Figure 8 (and on the colour generator site):

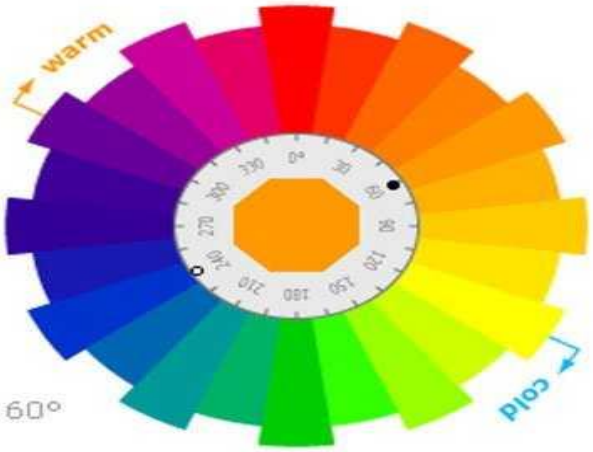


Fig. 3.3.8: Warm and Cool Colours.

Warm colours are those colours that would remind you of the summer, sun or fire. They consist of violets to yellows. Cool colours might remind you of spring, ice, or water. Those colours range from yellow-green back to violet. If you notice how the colours work on the wheel, you'll soon discover that you can't choose one colour

without choosing its opposite in "temperature." So, if you pick a hot red, the opposite is a cool green. Or, if you pick a cool blue-green, you'll end up with a spicy red-orange on the other side.

One example of a site that consistently uses a warm/cool colour combination is Ecolution, as seen in below.

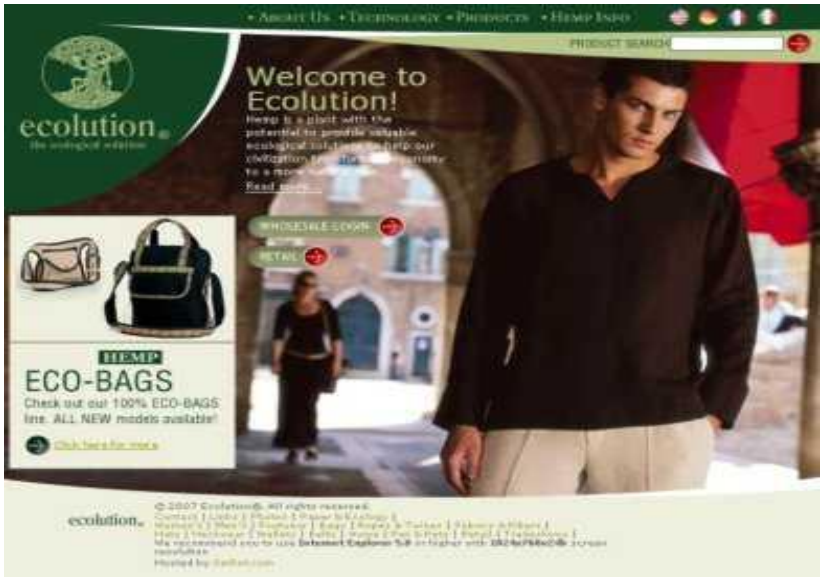


Fig. 3..3.9: Ecolution—a good example of warm/cool colours.

Ecolution usually uses red as an accent colour on their home page in contrast to their green logo. They then blend the two contrasting colours with varying tints, tones and shades of those two colours. Even the "blacks" in an image can lean toward "warm" or "cool," as do whites. Overall, the photograph is "warm," which plays nicely with the stark pure green. Although they use the same colours as Greenpeace, the Ecolution site takes on less "glare" with the rich tones and shades in the photograph.

You never realized that colour theory was so easy, did you? Well, then…let's complicate the issue a bit…

3.3.5 Triadic colour schemes

A triadic colour scheme is created when you pick one colour and then pick two other colours that lie equidistant from each other around the circle, like so:

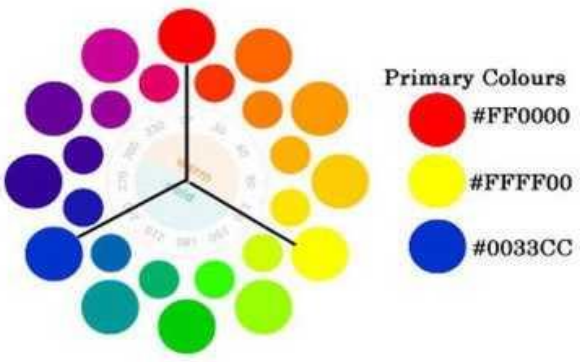


Fig. 3.3.10: A Triadic Colour Scheme.

I chose the primary colours for this scheme as I wanted to show how the colour schemes seem to contain a method to the madness. It's no accident that the primary colours lie where they do on the colour wheel, as each colour contains an equal amount of secondary and tertiary colours between each primary. But, a primary colour triadic scheme can seem old-hat as it has been overused. Instead, you might try some other colour choices at the online colour generator, something like the image below

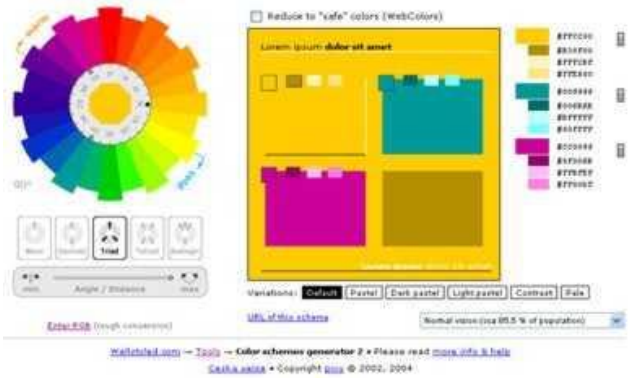


Fig. 3.3.11: An Alternative Triadic Colour Scheme.

The above triad scheme is built from orange-yellow, blue-green, and red-violet. I picked the orange-yellow first (notice the dark dot on the inner section of the colour wheel at left), and then chose the **Triad** selection located beneath the wheel. The generator automatically chose the triadic choices including all tints, tones and shades. The accompanying colours are marked on the colour wheel with the hollow dots, just as the complementary colour was noted in the monochromatic example.

Now, this is where a real colour wheel might come in handy, as the online results didn't quite match the results of a hand-held colour wheel. When I pushed the **Angle/Distance** tool under the colour wheel to "max," however, it seemed to match the colour wheel I held in my hand.

The results shown above are those that matched the colour wheel the closest.

The triadic colour scheme also contains warm and cool colours, but one temperature will predominate. Usually, the temperature that will overshadow the other is the one that you chose on the front end. In this case, I initially chose the orange-yellow, which is a warm colour. The warm colours shown above will predominate as a result, with one of the other two colours lending the cool contrast.

Puzzle Pirates—A good triadic colour scheme.

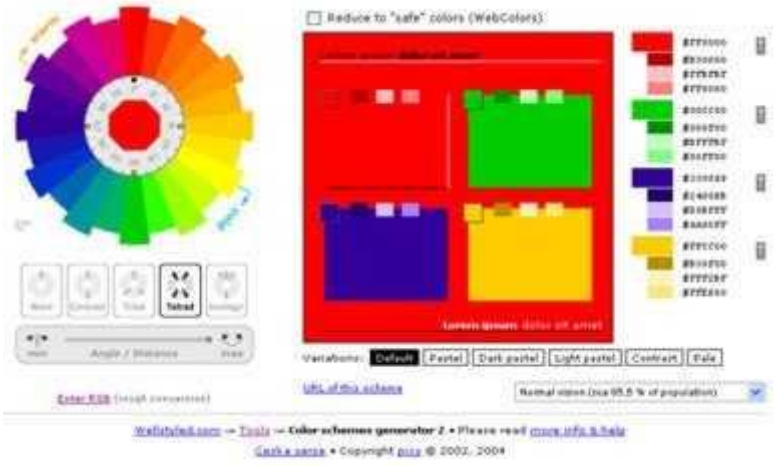
Puzzle Pirates, shown above, uses a triadic scheme on their home page. They use the primary red-blue-yellow scheme, and this primary scheme is perfect for a kid's game site. Note that the blue is predominant and that the reds and yellows are used as accents and to lead the eye around the page.

3.3.6 Tetradic Colour Schemes

The more colours you choose, the more complicated the colour scheme. However, one trick is to find a tint, tone or shade and stick to those regions across the board rather than mix pure colours and their tints, tones and shades. This method works well with a colour scheme such as the four-colour tetradic scheme. This scheme is just like the complementary scheme, only you use two complementaries that are equidistant.

Tetradic colour schemes.

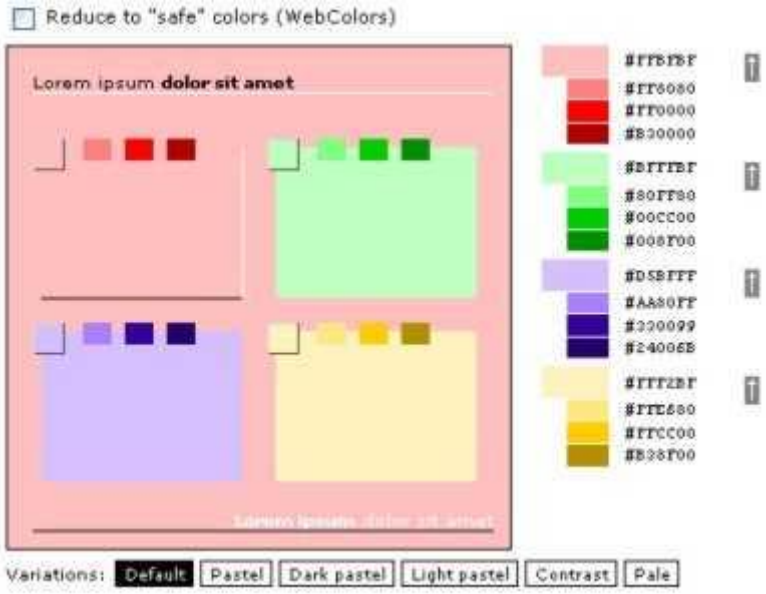
he Image below shows how a tetrad scheme works out online:



A tetradic colour scheme inside the online generator.

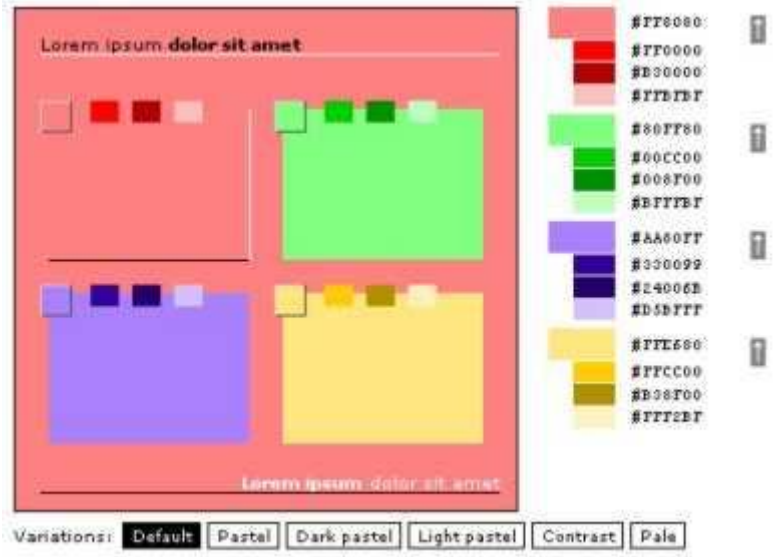
Note the black dot under the red in the colour wheel to the left. This was the first colour that I chose; I then clicked on the **Tetrad** button beneath the wheel. The four colours that showed up once again were a bit off from my hand-held colour wheel, but when I pushed the **Angle/Distance** tool under the colour wheel to "max," it seemed to match the colour wheel I held in my hand. The results shown above are those that matched the colour wheel the closest.

This colour scheme can become quite complicated, so what you might want to do at this point is to pick all four tints or tones or shades from the colours in the right column. You can make your choices by clicking the arrows at the far right. For instance, Figure 15 shows an example of a block filled with the tints of this colour scheme:



Tetradic tints.

An example of the mid-range tones:



Tetradic mid-range tones.

If you look closely at the squares above, you'll see that the generator also provides you with four monochromatic colour schemes. These schemes are shown both in the column to the right as well as in each square within the larger square.

The Jane Goodall Institute site—a good tetradic colour scheme example.

The Jane Goodall Institute site as shown above is one of the few sites that really carries the tetradic colour scheme well. Note the purple, the yellow tone, the red highlights in the photograph (the site holds more red further down on the page), and the greens. The purple doesn't match up exactly to the colour scheme generated by the online colour tool - it leans toward a red-violet - but it's close enough to use as an example of how you can use both a colour wheel and the online colour generator to produce ideas for your site.

Now, when you surf around the Web searching for colour and design ideas, keep your colour wheel close at hand to learn more about how designers use colour schemes on your favorite Web sites!

3.4 Animated GIFs

Animated GIF was established when basic animation was added to GIF89A spec of the Graphic Extension (GCE) which allows various frames in the file to be painted with time delays. Basically an Animated GIF is made up of frames with a number of frames that are displayed in succession each with it's own GCE.

Fig. 3.4.1: Example of an Image saved as .gif (did you notice how rough it looks?)

Self Assessment Questions (SAQ 1)

Self-Assessment Question(s)

1. What are the uses of the following
   - GIF
   - PNG
   - JPEG
2. What is the difference between Warm colours and cool colours?

Self-Assessment Answer

1. - GIF files for images that have a small, fixed number of colors. GIF files are always reduced to no more than 256 unique colors. The compression algorithm for GIF files is less complex than for JPG files, but when used on flat color images and text it produces very small file sizes

   - PNG supports 24 bit images and produces background transparency **without jagged edges**; however, some older web browsers do not support PNG images. PNG format supports RGB, Indexed Color, Grayscale, and Bitmap mode images.

   - The JPEG standard how an image is compressed into a stream of bytes and decompressed back into an image, but not the file format used to contain that stream. Jpeg compression algorithm is at its best when used on photographs and paintings or realistic scenes

2. Warm colours are those colours that would remind you of the summer, sun or fire. They consist of violets to yellows. Cool colours might remind you of spring, ice, or water. Those colours range from yellow-green back to violet.

3.5 Multimedia and interactivity

Multimedia

Mulitimedia comes in many formats such as images, animation, audio and video and it serves as a tool to embed audio and videos into a web page making it interactive and entertaing.

Social sites such as Facebook, twitter, youtube and tubenaija incorperate video and audio to entertain the user and give them more activities on the web site.

Browser Support

Before you learn how to embed videos and audio it is best you understand browser support. The first Internet browsers had support for text only, and even the text support was limited to a single font in a single color. Then came browsers with support for colors, fonts and text styles, and the support for pictures was added.

The support for sounds, animations and videos is handled in different ways by different browsers. Some elements can be handled inline, and some requires an extra helper program (a plug-in).A Plug-In can be used in HTML for many purposes. They can be used to display maps, verify your bank id, control your input, and much more.

Multimedia Formats

Multimedia elements (sound or video) are stored in media files.

The most common way to discover the media type is to look at the file extension. When a browser sees the file extensions .htm or .html, it will assume that the file is an HTML page. The .xml extension indicates an XML file, and the .css extension indicates a style sheet. Picture formats are recognized by extensions like .gif and .jpg.

Multimedia elements also have their own file formats with different extensions like .swf, .wmv, .mp3, and .mp4. there are many other formats available however these are the most commonly used audio and video formats on the internet.

Embedding Audio into your website

Let’s assume you have an audio file called horse.mp3 and horse.oga and you want to embed it into a page how can it be done? Take a look at these codes below:

```
<audio controls="controls" height="100" width="100">
  <source src="horse.mp3" type="audio/mp3" />
  <source src="horse.ogg" type="audio/ogg" />
<embed height="100" width="100" src="horse.mp3" />
```

```
</audio>
```

The first line sets the audio controls and features i.e the height and width

```
<audio controls="controls" height="100" width="100">
```

The second and third line determines the source of the file and the type

```
<source src="horse.mp3" type="audio/mp3" />
<source src="horse.ogg" type="audio/ogg" />
```

The whole essense of using thesame name but different file formats is allowing computers who do not have one file format plugin to be able to play the audio with another format.

The forth line embeds the height and width and also the sourse of the audio file.

```
<embed height="100" width="100" src="horse.mp3" />
```

The last line terminates the audio controls

```
</audio>
```

Embedding Video

Assuming you have a video clip named movie.mp4, movie.ogg, movie .webm, movie.mp4 and movie.swf and you want to embed it into a web page, the following codes below illustrate that.

```
<video width="320" height="240" controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  <source src="movie.webm" type="video/webm" />
  <object data="movie.mp4" width="320" height="240">
    <embed src="movie.swf" width="320" height="240" />
  </object>
</video>
```

The first line opens a video tag and specifies the width as 320 and the height as 240

```
<video width="320" height="240" controls="controls">
```

The next three lines illustrate the source of the movie and the type.

```
<source src="movie.mp4" type="video/mp4" />
<source src="movie.ogg" type="video/ogg" />
<source src="movie.webm" type="video/webm" />
<object data="movie.mp4" width="320" height="240">
<embed src="movie.swf" width="320" height="240" />
```

The essense of allowing different formats is to allow cross compatiblity with devices that can play the video with one format rather than another.

The fifth, sixth and seventh line allows for compartiblity with older browsers

```
<object data="movie.mp4" width="320" height="240">
  <embed src="movie.swf" width="320" height="240" />
 </object>
```

And finally the last line termainates the video tag

```
</video>
```

Self-Assessment Question(s)

1. Explain in your own words Animated Gifs.
2. Explain the term multimedia.
3. Explain the term Browser Support.

Self-Assessment Answer

1. Animated Gifs are a group of individual pictures tied together allowing various frames in the file to be shown with time delays.
2. Mulitimedia comes in many formats such as images, animation, audio and video and it serves as a tool to embed audio and videos into a web page making it interactive and entertaining.
3. Browser can be used to support colors, fonts and text styles, and the support for pictures can be added.

   The support for sounds, animations and videos is handled in different ways by different browsers. Some elements can be handled inline, and some requires an extra helper program (a plug-in).A Plug-In can be used in HTML for many purposes. They can be used to display maps, verify your bank id, control your input, and much more.

4.0 Conclusion

Images, color and Multimedia are critical part of your website design. If 65% of the population describes themselves as visual learners, then you have to plan for people who want to look or listen to instead of reading words alone.

5.0 Summary

In this study unit we have been able to discuss the following:

1. graphics GIF, JPEG and PNG formats,

2. design graphics with palette,
3. animated GIFs, multimedia and interactivity.

6.0 Tutor Marked Assignments

1. Explain in your own words Colours, tints, and shades.
2. Explain Monochromatic colour schemes in your own words.
3. Explain how to use complementary colours.
4. Why use Triadic colour schemes?

7.0 References/Further Reading

HTML Embedding Multimedia www.tutorialspoint.com

GIF, PNG, JPG. Which One To Use? http://www.sitepoint.com/gif-png-jpg-which-one-to-use/

Colour: www.w3schools.com

Colour Theory: http://www.artesmorit.com/color-theory-tetradic-color-schemes/

Module 3

Introduction to Other Web Software

Unit 1: Introduction to Javascript, DHTML and XML

Unit 2: XHTML, WAP and WML

Unit 1

Introduction To Javascript, Dhtml And Xml

Content

1.0 Introduction

Java script is the most commonly used client side scripting language. Just like CSS, JavaScript is incorporated into the HTML pages. Java script works with HTML to send its script to browsers hence it is the browser that utilizes the java script.

Java script is not the same as Java although they use similar rule, JavaScript is primarily a scripting language for use within HTML pages while Java is a programming language.

2.0 Learning Outcomes

At the end of this Unit, you should be able to:

i. Define of JavaScript and relationship between JavaScript and HTML
ii. Narrate the Historical background of JavaScript and browser incompatibilities
iii. Explain JavaScript tag, script in head element and body element
iv. Using an external JavaScript
v. Discuss the DHTML, XML and uses of XML
vi. Apply the XML syntax and structure, XML naming rules
vii. Use comments and different ways to modify text

3.0 Learning Content

3.1 Introduction to JavaScript

JavaScript is the scripting language that you use to add behavior to your web pages - it can be used to validate the data you enter into a form (tell you if it is in the right format or not), provide drag and drop functionality, change styles on the fly, animate page elements such as menus, handle button functionality, and a million other things. Most modern JavaScript works by finding a target HTML element, and then doing something to it, just like CSS, but the way it operates, the syntax etc is rather different.

JavaScript is a more complicated and expansive subject than HTML and CSS.

History of Javascript

JavaScript was developed by Brendan Eich, then working at Netscape, as a client side scripting language (even though there's no fundamental reason why it can't be used in a server side environment).

Originally the language was called Live Script, but when it was about to be released Java had become immensely popular.

Java and JavaScript both descend from C and C++, but the languages (or rather, their ancestors) have gone in quite different directions. You can see them as distantly related cousins. Both are object oriented (though this is less important in JavaScript than in many other languages) and they share some syntax, but the differences are more important than the similarities.

If you are a C++ or Java programmer you will be surprised by some of JavaScript's features.

JavaScript **is not** a programming language in strict sense. Instead, it is a scripting language because it uses the browser to do the dirty work. If you command an image to be replaced by another one, JavaScript tells the browser to go do it. Because the

browser actually does the work, you only need to pull some strings by writing some relatively easy lines of code. That's what makes JavaScript an easy language to start with.

Browser Incompatibilities

When a user receives a page which includes JavaScript, the JavaScript interpreter of his browser kicks in and tries to execute the script. Now the main problem here is that the various browsers each use their own interpreter, and that sometimes browser vendors have chosen not to implement a bit of JavaScript. Their reasons were usually related to business advantage over the competitors.

Hence the feared *browser incompatibilities*.

In addition each new browser version understands more JavaScript and allows more and more parts of the HTML page to be changed by scripts. This leads to even more incompatibilities.

It's best to solve compatibility problems on a case–by–case basis. In fact, most pages on this site have been written precisely because of browser incompatibilities.

JavaScript Versions

There have been several formal versions of JavaScript.

1.0: Netscape 2

1.1: Netscape 3 and Explorer 3 (the latter has *bad* JavaScript support, regardless of its version)

1.2: Early Version 4 browsers

1.3: Later Version 4 browsers and Version 5 browsers

1.4: Not used in browsers, only on Netscape servers

1.5: Current version.

2.0: Currently under development by Brendan Eich and others

The Script Tag

In other to insert JavaScript into an HTML Page, use the script tag <script> and </script> contain the JavaScript

```
<script>
        alert("My First JavaScript");
                </script>
```

You don't have to understand the code above. Just take it for a fact, that the browser will interpret and execute the JavaScript code between the <script> and </script> tags.

In other to apply this simple code into a web page, open a new page and add these contents

```
<!DOCTYPE html>
<html>
<body>
<h1>My first web page</h1>
<p id=“demo”> My First Web page </p>
document.getElementById(“demo”).innerHTML=”My First JavaScript”;
</script>
</body>
</html>
```

Writing Directly into the Document Output

The example below writes a <p> element directly into the HTML document output:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<script>
document.write("<p>My First JavaScript</p>");
</script>
</body>
</html>
```

Please not that if you use document. Write () directly into the document output. If you execute document.write after the document has finished loading, the entire HTML page will overwritten:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My First Paragraph.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
document.write("Oops! The document disappeared!");
}
</script>
```

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

Script in Head Element and Body Element

You can also place unlimited number of scripts in your web document. You can have scripts both in the body and in the head element at the same time.

```
<html>
        <head>
                <style>
Document.getElementbyId(“demo1”).innerHTML=”My Fist JavaScript”;
                </style>
                        </head>
                                <body>
                                        <h1>My first web page</h1>
                                                <p id=“demo”> My First
Paragraph </p>                                                  <p
id=“demo2”>My second Paragraph </p>

document.getElementById(“demo”).innerHTML=”My First JavaScript”;
                </script>
                                </body>
                                        </html>
```

Using an External JavaScript

You may have noticed that our previous HTML used our Java Script codes internally but just like you learnt in our CSS module you can also insert your rules externally.

Firstly open the HTML document and add these:

```
<!DOCTYPE html>

<html>
        <head> <title>My external JavaScript</title>
                        </head>
<body>
<script src="myScript.js"></script>

</body>
</html>
```

Now open a new page and add these into it

document.getElementById("demo").innerHTML="My First JavaScript";

Now save it in the same folder as the first code as myScript.js

One of the most popular use of JavaScript is to change images when the mouse hangs over a link.

The Mouseover

The following script is inserted into the document head, to cache the images, ensuring the rollover effect is fast:

```
<script type="text/javascript">
/************************************
preload images for use in the mouseover
************************************/
if( document.images ) { //check that changing images is supported
  var overVariableImg = new Image(); overVariableImg.src =
"LOCATION_OF_MOUSEOVER_IMAGE";
  var outVariableImg = new Image(); outVariableImg.src =
"LOCATION_OF_MOUSEOUT_IMAGE";
}
</script>
```

The image may be placed anywhere on the document (if the link is in a positioned element -<div style="position:absolute;" ...> Netscape 4 will nees the image to be in the same positioned element), but I have chosen to put the image as part of the link. The image is defined using:

```
<img src="LOCATION_OF_MOUSEOUT_IMAGE" name="IMAGE_NAME"
height="16" width="16" alt="" border="0">
```

And the link is defined using:

```
<a href="Whatever"
  onmouseover="if( document.images ) { document.images['IMAGE_NAME'].src =
overVariableImg.src; }"
  onmouseout="if( document.images ) { document.images['IMAGE_NAME'].src =
outVariableImg.src; }">
```

The Preloader

The preloader script is inserted into the document head and is available for you to use as a JavaScript header file. The preloader is designed to interact with progress bar scripts, such as my DHTML progress bar script which I have used here.

To download the script(s), see the script license, and check details like browser compatibility, use the links on the navigation panel at the top of this page.

Self-Assessment Question(s)

1. Discuss briefly, the history of JavaScript
2. List all the JavaScript versions that you know.
3. Discuss how to use an external JavaScript.

Self-Assessment Answer

1. JavaScript is the scripting language that you use to add behavior to your web pages—it can be used to validate the data you enter.
   JavaScript is incorporated into the HTML pages. Java script works with HTML to send its script to browsers hence it is the browser that utilizes the java script.
   2.
   - 1.0: Netscape 2
   - 1.1: Netscape 3 and Explorer 3 (the latter has *bad* JavaScript support, regardless of its version)
   - 1.2: Early Version 4 browsers
   - 1.3: Later Version 4 browsers and Version 5 browsers
   - 1.4: Not used in browsers, only on Netscape servers
   - 1.5: Current version.
   - 2.0: Currently under development by Brendan Eich and others
2. JavaScript can be inserted externally using the following tags
   <style>
   Document.getElementbyId(“demo1”).innerHTML=”My Fist JavaScript”;
   </style>
   It works just like that of inserting external style sheet for Cascade Style sheet i.e. CSS.

3.2 DHTML

DHTML or dynamic Hypertext Markup Language is a docment Object Model for HTML, it can also be refered as a standard programming interface for HTML. It is Platform and lanuguage independent. It uses W3C standard.

HTML DOM

HTML DOM defines the objects and properties of all HTML elements, and the methods or interface to access them.

In other words HTML DOM can be referred as how to get, change, add, or delete HTML elements

3.3 XML

INTRODUCTION

Xml or extended Markup Language is a markup language. It uses tags just like in HTML to label, categorize and organise information and data in a way that is sutiable to the programmer.

Contest like text and images are all part of the codes that the mark up tags contain.

With XML, you can send the same information to various locations i.e. opening a web page on a Personal Computer and opening the same page on a mobile phone and get similar results.

In XML you can also create your own tags to suit your data and document needs.

XMl was not specificcally designed only for Web pages, if you try to open a XMl file on your browser you will notice that there is not much to look at.

One of the most common misconceptions about XML is that it is a programming language. Even though XML is used with other programming languages like Java for certain types of application development it is a Markup language just like HTML and WML and not a programming language.

Uses of XML

There are many uses of XML these include

XML is used to separate data from HTML: Computer systems and databases contain data in incompatible formats. However XML data is stored in plain text format. This allows the database to be easily accessible.

XML simplifies Data Transport: Many computer systems use different database structures across the internet and may be incompatible and time consuming to resolve the issues. XML resolves these issues since the data can be linked by incompatible applications.

XML makes Data more available: Different applications can access your data, not only in HTML pages, but also from XML data sources. With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities.

XML is used to Create New Internet Languages: A lot of new Internet languages are created with XML.

- Here are some examples:
- XHTML
- WSDL for describing available web services
- WAP and WML as markup languages for handheld devices
- RSS languages for news feeds
- RDF and OWL for describing resources and ontology
- SMIL for describing multimedia for the web

- The benefits of using XML
- XML is brimming with benefits. Here’s a list that puts these benefits

- **Unlimited elements:** You get to create your own elements and attributes
- Instead of working with a restricted, predefined set.
- **Structured data:** Applications can extract information that they need from XML documents.
- **Data exchange:** Using XML enables you to exchange database contents or other structured information across the Internet or between dissimilar applications.
- **XML complements HTML:** XML data can be used in HTML pages.
- **XML documents are well formed:** XML documents must follow certain rules. This consistency makes such documents easier to read and create.
- **Self-describing:** No prior knowledge of an XML application is needed. Knowing HTML can really help you understand more about XML (but software thrives on self-describing documents and data).
- **Search engines:** XML delivers a noticeable increase in search relevance because it provides ample contextual information and explicit labels for document elements.
- **Updates:** No need to update an entire site page by page; the Document Object Model (DOM) built into XML documents permits individual elements to be accessed (and updated).
- **User-selected view of data:** Different users can access different information or can present the same information in various ways.

XML Syntax and Structure

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

<note>
        <to>Tove</to>
        <from>Jani</from>
                <heading>Reminder</heading>
                 <body>Don't forget me this weekend!</body>
                        </note>
```

The first line contains the xml version type or XML type. It defines the XML version used.

The next line is the root element of the document <note>

The next 4 lines describe 4 child elements of the root (to, from, heading, and body):

```
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
```

And finally the last line defines the end of the root element:

```
</note>
```

Structure

XML documents must contain a root element. This element is "the parent" of all other elements.

The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree.

All elements can have sub elements (child elements):

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

The terms parent, child, and sibling are used to describe the relationships between elements. Parent elements have children. Children on the same level are called siblings (brothers or sisters).

All elements can have text content and attributes (just like in HTML).

A good illustration of this structure is as below:

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
```

```
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

XML Naming Rules

XML elements must follow these naming rules:

i. Names can contain letters, numbers, and other characters
ii. Names cannot start with a number or punctuation character
iii. Names cannot start with the letters xml (or XML, or Xml, etc)
iv. Names cannot contain spaces
v. Any name can be used, no words are reserved.

Best Naming Practices

Make names descriptive. Names with an underscore separator are nice: <first_name>, <last_name>.

Names should be short and simple, like this: <book_title> not like this: <the_title_of_the_book>.

Avoid "-" characters. If you name something "first-name," some software may think you want to subtract name from first.

Avoid "." characters. If you name something "first.name," some software may think that "name" is a property of the object "first."

Avoid ":" characters. Colons are reserved to be used for something called namespaces (more later).

XML documents often have a corresponding database. A good practice is to use the naming rules of your database for the elements in the XML documents.

Non-English letters like éòá are perfectly legal in XML, but watch out for problems if your software vendor doesn't support them.

Self-Assessment Question(s)

1. Highlight the uses of XML.
2. List the XML naming rules.
3. Discuss the best naming practices in XML.

Self-Assessment Answer

4.0 Conclusion

In all using XML, JavaScript and DHTML have advantages because they give the website or web page user intractability they also require the programmer use less codes.

They are also be used with mobile devices.

5.0 Summary

In this Unit 1, you have learnt the following aspects:

i. Definition of JavaScript and relationship between JavaScript and HTML
ii. History of JavaScript and browser incompatibilities
iii. JavaScript tag, script in head element and body element
iv. Using an external JavaScript
v. DHTML, XML and uses of XML
vi. XML syntax and structure, XML naming rules

6.0 Tutor Marked Assignments

1. Discuss script in head element and body element
2. Write XML syntax and structure
3. Use XML structure to create a xml document where Food is the root' Fruits is the child and Mangoes oranges, bananas, pears, and Water Melons are sub-children
4. Write briefly on DHTML
5. Discuss JavaScript tag

7.0 References/Further Reading

Introduction to JavaScript: www.w3schools.com

JavaScript Tutorial www.howtocreate.co.uk

Introduction to XML: http://www.w3schools.com/xml/xml_whatis.asp

Introduction to DHTML: http://www.w3schools.com/dhtml/dhtml_intro.asp

Unit 2

XHTML, WAP and WML

Content

1.0 Introduction

HTML as we know and love it has for many reasons become rather lax and unruly. If you've diligently tested your Web pages on different browsers only to find that your carefully crafted masterpiece looks great in IE5x, but becomes an illegible monster in Netscape 4x.

Well, we could spend all of our time whining about browser conformity, proprietary tags and standards. Or, we could take a pro-active stance and support the World Wide Web Consortium's first recommendation for XHTML: XHTML 1.0

This study unit takes a "quick start" approach aimed at the HTML author who wants to further their skills, and it concludes with links to more detailed information in WAP and WML

2.0 Learning Outcomes

At the end of the Study Unit you should have good knowledge about the following:

i. XHTML, different between HTML and XHTML and more,
ii. History of WAP,
iii. WAP internet model,
iv. History of WML,
v. How to create link in WML,
vi. Editing text with WML.

3.0 Learning Content

3.1 XHTM

According to the W3C:

"XHTML 1.0 is a reformulation of HTML 4.01 in XML, and combines the strength of HTML 4 with the power of XML.

XHTML 1.0 is the first major change to HTML since HTML 4.0 was released in 1997. It brings the rigor of XML to Web pages and is the keystone in W3C's work to create standards that provide richer Web pages on an ever increasing range of browser platforms including cell phones, televisions, cars, wallet sized wireless communicators, kiosks, and desktops."

Sound good so far? Then read on…

If you've no experience with XML you could be forgiven for being a little intimidated by it. But

if you can code your pages in HTML, you'll be pleased to know that learning XHTML will be extremely easy. It will also provide you with a superb introduction to XML along the way.

Essentially XHTML is just a stricter version of HTML 4.01 with a few considerations that you should be aware of as you mark up your pages.

As you may know, the eXtensible Markup Language is not a markup language at all, but a way of defining markup languages by use of a Document Type Definition, or DTD. XHTML is one such language and there are three different DTDs to choose from.

i. Strict – disallows use of all deprecated tags and attributes such as the <font> tag.

ii. Transitional – is far more forgiving and supports all those deprecated yet browser supported tags you most likely use every day.

iii. Frameset – is exactly the same as the transitional DTD but replaces the document body with frame attributes.

You'll probably want to use the transitional DTD as it provides the most forgiving environment for an introduction to XML and XHTML.

The Main Differences Between Html And Xhtml

The specification requires that your documents be "well formed", which means that you have to pay special attention to certain aspects of your code. Below are the key points you need to be aware of.

1. Nested elements

Firstly you need to tidy up the way you treat your page elements. XHTML does not tolerate incorrect nesting so something like this:

I'd probably have gotten away with it too if it weren't for

 you pesky W3C folks

Won't pass muster at the W3C's Validation service but

Buffy rules!
will be just fine. The same applies to all your markup tags.

2. Case Sensitivity

Both tags and their attributes are case sensitive in XHTML. The simple and strict rule is that all tags and attributes must be written in lower case. For example,

Some page

1. will get you roasted alive by the XHTML Validator, but
2. Well formed page

will work perfectly.

End Tags

Most HTML designers leave out the end tags to certain elements such as If you didn't know even had an end tag, you're not alone. Here are the tags most likely to catch you out: <th> <tr> <td>

What about images and line breaks?

Good question. These elements are similar, and all require an end tag. That's the way XML works, and of course XHTML is no exception even if there is no end tag in the HTML equivalent. You deal with this by including the end tag in its opener. Here's an example:

1. XHTML is strict but not really hard
2.

3. See what I mean?
4. <hr />

The trick is to leave a space before the closing tag so as not to confuse non-XHTML browsers.

Attributes

There are a couple of things you should be aware of when you're dealing with attributes. The first is that all your attributes must be enclosed within "double-quotes".

The second is that for those attributes that in HTML have no value such as <ul compact> you must specify one. It's done like this:

1. <ul compact="compact">

Other attributes to watch for are:

1. ismap="ismap"
2. declare="declare"
3. nowrap="nowrap"
4. compact="compact"
5. noshade="noshade"
6. checked="checked"

Special characters

I hate to say it, but this is the point where XHTML becomes a bit of a pain. Most of the above is just a matter of disciplining yourself and developing good coding habits, but there are a few problems here that require special mention. They'll almost certainly cause you trouble if you're unprepared!

1. XHTML can be a little problematic in its handling of <, > and & characters in CSS and JavaScript. XML browsers may remove your comments and thus your commented CSS. Use External stylesheets and JS scripts to be certain (although I've had no problems so far and do not do this on my own site).

2. Ampersands can be a problem within attributes as well. As a general rule of thumb, just use the corresponding HTML entities for &, <, > characters and make sure that you validate your pages properly.

Use Id Instead Of Name

The name attribute is now deprecated in favour of the new and prefered **id** attribute. Although it's supported, you'll get warnings when it comes to validation if you use name on a **map** tag, for instance.

A Simple Xhtml Document

Okay, enough with the do's and don'ts. If you're eager to get going here's a simple XHTML document to get you started.

```
1.  <?xml version="1.0" encoding="UTF-8"?>  <br>
2.  <!DOCTYPE html PUBLIC  <br>
3.          "-//W3C//DTD XHTML 1.0 Transitional//EN"  <br>
4.          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <br>
5.  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">  <br
    >
6.  <head>  <br>
7.  <title>XHTML is easier than you thought!</title>  <br>
8.  </head>  <br>
9.  <body>  <br>
10. <hr />  <br>
11. <p>As long as you remember the rules and guidelines   <br>
12. above<br />   <br>
13. you'll soon be writing well formed documents.  <br>
14.  No really, you will! </p>  <br>
15. <hr />  <br>
16. </body>  <br>
17. </html>
```

A detailed explanation of the declarations at the top of the document is beyond the intended scope of this "quick start" guide (and quite unnecessary for most designers), but here's the simple version.

Lines 1 Tells the browser that we're using XML 1.0 and gives its encoding as 8-bit Unicode.

Line 2-4 States the DTD we're using, which in this case is the transitional version.

Line 5 Declares in the <html> tag the XHTML name space and language attributes.

And there you have it, you're all set to start writing well-formed, standards-compliant XHTML pages! All you need do is use the code above as your basic template and start getting into good coding habits from the outset. If you find that you can't validate every page on your site properly then don't worry, it's a pretty tough call. As long as you're making an effort to validate as much as possible you're doing a good job.

Self-Assessment Question(s)

1. Highlight the uses of XML.
2. List the XML naming rules.

Self-Assessment Answer

1.
- XML is used to separate data from HTML
- XML simplifies Data Transport.
- XML makes Data more available.
- With XML, your data can be available to all kinds of "reading machines"

2.
- Names can contain letters, numbers, and other characters
- Names cannot start with a number or punctuation character
- Names cannot start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces

3.2 WAP

WAP is the de facto worldwide standard for providing Internet communications and advanced telephony services on digital mobile phones, pagers, personal digital assistants and other wireless terminals - *WAP Forum*

WAP stands for **W**ireless **A**pplication **P**rotocol. Per the dictionary definition for each of these words, we have:

a. **Wireless:** Lacking or not requiring a wire or wires: pertaining to radio transmission.

b. **Application:** A computer program or piece of computer software that is designed to do a specific task.

c. **Protocol:** A set of technical rules about how information should be transmitted and received using computers.

History of Wap

The Wireless Application Protocol (WAP) is a result of joint efforts taken by companies teaming up in an industry group called WAP Forum (www.wapforum.org).

On June 26 1997 Ericsson, Motorola, Nokia, and Unwired Planet took the initiative to start a rapid creation of a standard for making advanced services within the wireless domain a reality. In December 1997 WAP Forum was formally created, and after the release of the WAP 1.0 specifications in April 1998, WAP Forum membership was opened to all.

The WAP Forum now has over 500 members and represents over 95 percent of the global handset market. Companies such as Nokia, Motorola and Ericsson are all members of the forum.

The objective of the forum is to create a license-free standard that brings information and telephony services to wireless devices.

Though WAP is a new technology it uses some concepts found on the internet. This enables programmers easily get used to the structure of WAP.

Exampls of concepts it uses include WML Script, Wireless Telephony Application Interface and Otimized Protocol stack.

The Internet Model vs The WAP Mobile

The Internet Model:

The 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.

The figure below shows how a WWW 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.

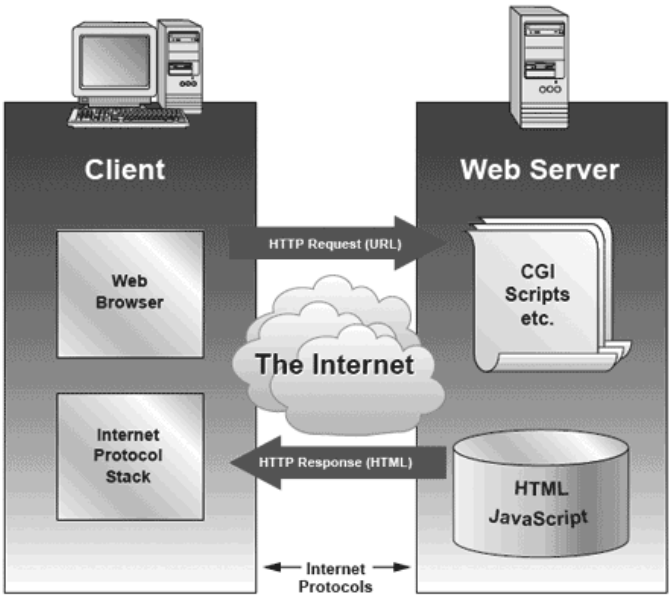


Fig. 3.2.1: The Wap Model

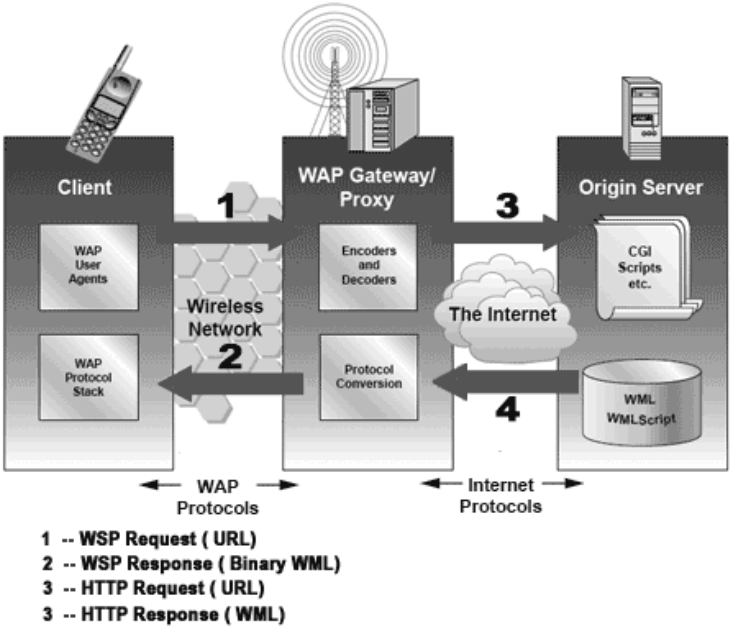


Fig. 3.2.2: WAP Programming Model

The illustration above shows the WAP programming model. Note the similarities with the Internet model. Without the WAP Gateway/Proxy the two models would have been practically identical.

A markup language - the Wireless Markup Language (WML) has been adapted to develop optimized WAP applications. In order to save valuable bandwidth in the wireless network, WML can be encoded into a compact binary format. Encoding WML is one of the tasks performed by the WAP Gateway/Proxy

Requirements

At minimum, developing WAP applications requires a web server and a WAP simulator. Using simulator software while developing a WAP application is convenient as all the required software can be installed on the development PC.

Although software simulators are good in their own right, no WAP application should go into production without testing it with actual hardware. The following list gives a quick overview of the necessary hardware and software to test and develop WAP applications:

i. a Web server with connection to the Internet
ii. a WML to develop WAP application
iii. a WAP simulator to test WAP application
iv. a WAP gateway
v. a WAP phone for final testing

Microsoft IIS or Apache on Windows or Linux can be used as the web server and Nokia WAP Toolkit version 2.0 as the WAP simulator.

WAP supports different types of MIME (Multi-Purpose Internet Mail Extension) types or File types these include WML (.WML) WMLScript (.wmls) WBMP (.wbmp).

The essence of MIME is to ensure that in the WAP architecture, the web server communicates with the WAP gateway accepting requsts and returning WML code to the gateway.

Self-Assessment Question(s)

1. Discuss the best naming practices in XML.

Self-Assessment Answer

- Non-English letters like éòá are perfectly legal in XML, but watch out for problems if your software vendor doesn't support them.
- Avoid "." characters. If you name something "first.name," some software may think that "name" is a property of the object "first."
- XML documents often have a corresponding database. A good practice is to use the naming rules of your database for the elements in the XML documents.
- Make names descriptive. Names with an underscore separator are nice: <first_name>, <last_name>.
- Names should be short and simple, like this: <book_title> not like this: <the_title_of_the_book>.

- Avoid "-" characters. If you name something "first-name," some software may think you want to subtract name from first.

3.3 WML

WML (Wireless Markup Language) is the new web language for making sites on mobile phones. Over the past few months new WAP (Wireless Applications Protocol) phones have become extremely popular and many large websites have created special ‘mobile’ versions of their site. Many people predict that, over the next few years, WAP sites will become extremely popular and e-commerce over mobile phones will be widly available.

The syntax fo WML is as follows:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<p> Welcome to my first page </p>
</wml>
```

The first two lines show the xml version and Document type.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.xml">
```

The next line is the body in which all the wml codes will lie.

```
<wml>
<p>Welcome to my first page</p>
</wml>
```

To save the page, it must be a .xml file type. If you want to preview it, i would recommend using Adobe Dreamweaver / Adobe Device Central,in other to have a real feel of what this page looks on a mobile emulator.

Fig. 3.3.1: Mobile Emulator

Cards

The card element is the basic unit of content in WML and the parent of all lower-level elements in the document; that is to say, it is the only child of the wml element. So, one card and a small snippet of text is all it takes to get you to the tiny silver screen. The most practical WML documents consist of multiple cards

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org/DTD/wml13.dtd">
<wml>
<card>
<p>Hello !!!</p>
</card>
<card>
<p>Welcome to a new page</p>
</card>
</wml>
```

Line Break

 is the line breaking tag in WML, which is the same as that in HTML. The following WML example demonstrates the usage of line breaks:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org/DTD/wml13.dtd">
<wml>
  <card id="card1" title="Line Break">
    <p>
      Before br<b><br/></b>After br
    </p>
  </card>
</wml>
```

Pre tag

The <pre> tag is used to specify preformatted text in WML. Preformatted text is text of which the format follows the way it is typed in the WML document.

In WML, leading and trailing whitespaces (i.e. spaces, tabs and newlines) of a paragraph are not displayed. Furthermore, two or more whitespaces in a paragraph are shown as one whitespace on the screen of mobile devices. Such behavior is demonstrated in the following WML example.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org/DTD/wml13.dtd">
<wml>
  <card id="card1" title="WML Tutorial">
    <p> Hello, welcome
to
our
        WML tutorial .</p>
  </card>
</wml>
```

The result of the above WML example in WAP browsers is shown below

To preserve the formatting of the text, you have to use the <pre> element. WAP browsers treat the content enclosed in the <pre></pre> tag pair as preformatted and whitespaces are left intact.

The following WML example illustrates how to use the <pre> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org/DTD/wml13.dtd">
<wml>
  <card id="card1" title="WML Tutorial">
    <pre> Hello, welcome
to
our
        WML tutorial .</pre>
  </card>
</wml>
```

The result of the above WML example in WAP browsers is shown below

Editting Text with WML

Even though wireless device screens are typically somewhat limited in terms of their graphical capabilities, it is possible to format text in WML documents to a certain extent. The good news is that if you've ever formatted text in HTML, you'll find the WML approach to be very familiar. Following are the text formatting elements used in WML:

<b> Bolds text
<big> Enlarges text relative to the browser's default text size
 Emphasizes text in a browser-defined fashion
<i> Italicizes text
<small> Reduces text relative to the browser's default text size
 Renders text "strong," whatever that means to the given browser.
Putting all these together we have this

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

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
   "http://www.wapforum.org/DTD/wml13.dtd">
    <wml>
    <card>
    <p><b>Welcome</b></p>
         <p><big>FilmTime!</big></p>
    <p><em>FilmTime!</em></p>
    <p><i>FilmTime!</i></p>
    <p><small>FilmTime!</small></p>
    <p><strong>FilmTime!</strong></p>
  </card>
 </wml>
```

Inserting Links And Images With Wml

Links

A WML card can be set up to display the anchor functions of WML.

<anchor>

The <anchor> tag always has a task ("go", "prev", or "refresh") specified. The task defines what to do when the user selects the link. In this example, when the user selects the "Next page" link, the task says "go to the file test.wml":

Take a look at this illustration:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card title="Anchor Tag">
<p>
<anchor>Next page
<go href="test.wml"/>
</anchor>
</p>
</card>
</wml>
```

<a>

The <a> tag always performs a "go" task, with no variables. The example below does the same as the <anchor> tag example:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card title="A Tag">
<p>
<a href="test.wml">Next page</a>
</p>
</card>
</wml>
```

Images

A WML card can be set up to display an image:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
```

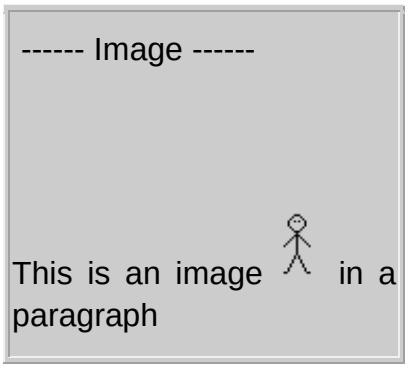
<card title="Image">

This is an image

in a paragraph

</card>

</wml>



Note: .wbmp is the only image type that cn be displayed in a WAP browser.

Self-Assessment Question(s)

1. Write briefly on the word WAP.
2. What is WML?
3. Discuss editing text with WML.

Self-Assessment Answer

1. WAP stands for Wireless Application Protocol and it is the de-facto standard for providing internet communication and advanced telephony services on digital mobile phones. The wireless application protocol is the result of joint efforts taken by Ericsson, Motorola, Nokia and Unwired Planet teamed creating an Industry group called WAP Forum. The objective of the forum is to create a license free standard that brings information and telephony services to wireless devices
2. WML stands for Wireless Markup Language and it is the web language for making sites on mobile phones.
3. The syntax for WML is as follows:

   <?xml version="1.0"?>

   <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"

   "http://www.wapforum.org/DTD/wml_1.1.xml">

```
<wml>
<p> Welcome to my first page </p>
    </wml>
```

4.0 Conclusion

In today's society there is a growing demand for not just websites but interactive mobile sites. WAP is the worldwide standard for providing Internet communications on digital mobile phones. WML is the de-facto language used in creating Mobile Web sites. It is valuable today because many access the internet via mobile devices rather than using conventional Personal Computers. These mobile devices require less bandwidth and graphics but must be as interactive as a regular web site.

5.0 Summary

In this study unit we have been able to discuss the following:

1. Brief history of WAP
2. WAP internet model
3. Brief history of WML
4. How to create link in WML
5. Editing text with WML

6.0 Tutor Marked Assignments

1. List the necessary hardware and software needed to develop and test WAP application
2. Discuss the differences between HTML and xHTML
3. Discuss how to create link in WML
4. Write briefly on WAP internet model

7.0 References/Further Reading

WAP Introduction: http://www.tutorialspoint.com/wap/wap_introduction.htm

Introduction to WML and WMLScript:
http://www.webreference.com/js/column61/4.html

Creating WML Documents:
http://www.brainbell.com/tutorials/XML/Creating_WML_Documents.htm