Mt Kenya University

P.O. Box 342-01000 Thika

Email: info@mku.ac.ke

Web: www.mku.ac.ke

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: BIT 2105

COURSE TITLE: INTRODUCTION TO WEB DESIGN AND DEVELOPMENT

Instructional manual for BBIT – Distance Learning

TABLE OF CONTENTS

COURSE OUTLINE

BIT 2105: INTRODUCTION TO WEB DESIGN AND DEVELOPMENT

Pre-requisite: BIT 1102 - COMPUTER APPLICATIONS

BIT 1103 - COMPUTER ARCHITECTURE

Purpose: To enable the learner design and develop dynamic websites

Objectives: By the end of the course unit, the learner will be able to: -

- Describe the methodology for developing an organizational website.
- Define advanced structures using a mark-up language
- Enumerate the basic principles of web design.

Assessments: Continuous Assessment Tests (CATs) (30%), End of semester examination (70%), Total = 100%

Required text books

Niederst J., Learning Web design, Shroff Publishers

Text book for further reading

Peter, K., Web design, Tools and Techniques, Peach Pit press

BIT 2105: INTRODUCTION TO WEB DESIGN AND DEVELOPMENT - TOPICS – Details

Week 1: Introduction

- Services provided by the internet
- Important components of the web
- Common internet protocols
- Internet architecture

Week 2: Web project Management

- Project management
- Online project management systems
- Choosing a Web development firm
- Budget Planning for the Web site development project

- Cost estimation for web projects
- How to measure the size of the website

Week 3: Website interface design

- Website Design Basics
- Principles of User Interface Design
- Key Elements of a Good Website
- Web Site Organization
- Website wireframe

Week 4-5: HTML Page Layout Design

- HTML Forms
- HTML Iframes
- HTML Layouts

Week 6-8: Page Layout: Introduction to CSS (Cascading Style Sheets)

- Introduction to CSS
- CSS Syntax
- CSS Id and Class
- CSS Backgrounds Styling
- CSS Text Styling
- CSS Fonts Styling
- CSS Links Styling
- CSS Lists Styling
- CSS Tables Styling
- CSS Box Model
    - CSS Border
    - CSS Outline
    - CSS Margin
    - CSS Padding

Week 9-12: Client side scripting: Introduction to JavaScript

- Introduction to JavaScript

- JavaScript Statements
- JavaScript Comments
- JavaScript Variables
- JavaScript Operators
- JavaScript Conditional statements
- JavaScript Functions
- JavaScript Loops
- JavaScript Events
- JavaScript Try...Catch
- JavaScript Throw
- JavaScript Validation

Module compiler: David Kibaara

CHAPTER ONE: INTRODUCTION TO WEB DESIGN AND DEVELOPMENT BASIC CONCEPTS

Learning Objectives:

By the end of this chapter the learner shall be able to;

i. Explain the services provided by the internet

ii. Explain the different internet protocols

iii. Explain the advantages and disadvantages of client/server architecture.

1.1 Services Provided by the Internet

Electronic Mail - E-mail, also known as electronic mail, is one of the most popular Internet services. E-mail allows you to send messages to one person, or to send a message simultaneously to a group of people. One of the greatest advantages of e-mail over other forms of communication is the convenience to the recipient. Messages wait in your mailbox until you open it. Another advantage of an Internet e-mail account is that you can check your e-mail from any location with an internet connection.

FTP (File Transfer Protocol)-This facility is a method of gaining limited access to another machine in the Internet, and obtaining files from it. You need full Internet connectivity, to do ftp interactively. FTP has many advantages, for example, it allows you to get new free software, or updated versions of old programs, as well as useful data for your research. The most common way of using FTP is via **anonymous FTP**. When you start an ftp connection, you will be asked for a user name and a password.

Telnet: logging in to Remote Network Computers - Telnet is the Internet facility that allows you to execute commands on a remote host (another computer, most likely one to which you do not have physical access) as if you were logged in locally. You need to know the name of the machine to which you want to connect, and to have a valid user name in it. There is no such thing as "anonymous" telnet.

Usenet Newsgroups Usenet newsgroups, also called bulletin boards, are a similar e-mail conferencing system, but are less intrusive to the subscriber than list serves since messages are posted to Usenet sites around the world instead of appearing in each subscriber's mailbox. Usenet refers to the huge collection of messages which are posted to tens of thousands of newsgroups worldwide. Millions of people around the world regularly read newsgroup messages, following their favourite topics of interest. New newsgroups are added and old ones deleted every day. Usenet can provide a unique information resource not readily accessible from any other source. If you are looking for personal anecdotes about products, especially computer-related hardware and software products, how-to information, practical advice, or the latest news stories, newsgroup archives may be a valuable resource.

Internet Chat

Communication on the Internet goes even further than personal e-mail, newsgroups and mailing lists, to encompass real-time conversations (synchronous communication) among two or more people. Chat is available on the Internet through Internet Relay Chat or IRC. It consists of thousands of chat channels, each covering a different topic and with participants from all over the world.

Web Conferencing Many institutions are discovering new ways to integrate Internet communications into their organizations. One of the most popular ways is through the use of web or online conferencing.

Web conferencing is currently being used by businesses for employee training, meetings and general communication. Educational institutions are using web conferencing as a way to enhance on-site classes or distance education classes. Web conferencing is a tool which provides a way for "students" to share information, ask questions, get answers, discuss problems and work collaboratively. Conferencing provides opportunities to solve issues by providing a dynamic exchange of text, graphics, HTML links to information, audio, and video in a structured conversation organized by topic. Web conferences may take place in "real-time" where all participants are communicating at the same pre-arranged time.

1.2 Requirements for connecting to the internet

Internet service provider – an internet service provider provides you with a connection to the internet and the software you will need to navigate. An Internet service provider (ISP) is a company that provides access to the Internet. Access ISPs directly connect customers to the Internet using copper wires, wireless or fiber-optic connections.

Telecommunication line (Mobile or Wired) – a telephone line is required to connect you to the internet service provider.

Modem – a modem converts a digital signal received from a computer into an analogue signal that can be sent along ordinary telephone lines, and back to digital at the other end.

Web browser – a web browser is software used to view and download Web pages and various types of files such as text, graphics and video. Examples are Microsoft Internet Explorer or Netscape Navigator.

1.3 Important Components of the Web

Web Server

Web server can refer to either the hardware (the computer) or the software (the computer application) that helps to deliver content that can be accessed through the Internet. The most common use of Web servers is to host Web sites but there are other uses like data storage or for running enterprise applications. A web server is a computer programs that delivers (serves) content, such as web pages, using the Hypertext Transfer Protocol (HTTP), over the World Wide Web.

A **web search engine** is designed to search for information on the World Wide Web and FTP servers. The search results are generally presented in a list of results and are often called *hits*. The information may consist of web pages, images, information and other types of files. Some search engines also mine data available in databases or open directories. Unlike Web directories, which are maintained by human editors, search engines operate algorithmically or are a mixture of algorithmic and human input. Examples of search engines are yahoo, google, msn etc

Web crawler is an automated Web browser which follows every link it sees. The contents of each page are then analyzed to determine how it should be indexed (for example, words are extracted from the titles, headings, or special fields called meta tags). Data about web pages are stored in an index database for use in later queries. Some search engines, store all or part of the source page (referred to as a cache) as well as information about the web pages, whereas others, store every word of every page they find. When a user enters a query into a search engine the engine examines its **index** and provides a listing of best-matching web pages according to its criteria, usually with a short summary containing the document's title and sometimes parts of the text. Exclusions from web crawler can be made by the use of robots.txt.

Domain Name System (**DNS**) is a hierarchical naming system for computers, services, or any resource connected to the Internet or a private network. It associates information with domain names assigned to each of the participants. Most importantly, it translates domain names meaningful to humans into the numerical (binary) identifiers associated with networking equipment for the purpose of locating and addressing these devices worldwide.

The Domain Name System makes it possible to assign domain names to groups of Internet users in a meaningful way, independent of each user's physical location. Internet domain names are easier to remember than IP addresses such as 208.77.188.166 (IPv4) or 2001:db8::1f70:6e8 (IPv6). The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating authoritative name servers for each domain. Authoritative name servers are assigned to be responsible for their particular domains, and in turn can assign other authoritative name servers for their sub-domains. The DNS database of Domain names and the corresponding IP addresses can not be held on one machine. As a truly distributed resource, it is maintained by many organisations, each manages a little bit of it. DNS defines a tree structure, and each node on the tree is owned by one of the naming authorities. The owner of a node can create any number of child nodes, but each must have a unique name.

URLs Addresses for web sites are called URLs (Uniform Resource Locators). Most of them begin with http (HyperText Transfer Protocol), followed by a colon and two slashes. For example, the URL for the Florida Centre for Instructional Technology is http://fcit.usf.edu/ . 20
Some of the URL addresses include a directory path and a file name. Consequently, the addresses can become quite long. For example, the URL of a web page may be: http://fcit.usf.edu/holocaust/default.htm. In this example, "default.htm" is the name of the file which is in a directory named "holocaust" on the FCIT server at the University of South Florida.

Top-level domain Each part of a domain name contains certain information. The first field is the host name, identifying a single computer or organization. The last field is the top-level domain, describing the type of organization and occasionally country of origin associated with the address.
Top-level domain names include:

.com	Commercial
.edu	Educational
.gov	US Government
.int	Organization
.mil	US Military
.net	Networking Providers
.org	Non-profit Organization

Domain name country codes include, but are not limited to:

.au	Australia
.de	Germany
.fr	France

.nl	Netherlands
.uk	United Kingdom
.us	United States
.ke	Kenya

Routers

All of these networks rely on Network Access Point (NAPs), backbones and **routers** to talk to each other. What is incredible about this process is that a message can leave one computer and travel halfway across the world through several different networks and arrive at another computer in a fraction of a second! The routers determine where to send information from one computer to another. Routers are specialized computers that send your messages and those of every other Internet user speeding to their destinations along thousands of pathways. A router has two separate, but related, jobs:

- It ensures that information doesn't go where it's not needed. This is crucial for keeping large volumes of data from clogging the connections of "innocent bystanders."
- It makes sure that information does make it to the intended destination.

In performing these two jobs, a router is extremely useful in dealing with two separate computer networks. It joins the two networks, passing information from one to the other. It also protects the networks from one another, preventing the traffic on one from unnecessarily spilling over to the other. Regardless of how many networks are attached, the basic operation and function of the router remains the same. Since the Internet is one huge network made up of tens of thousands of smaller networks, its use of routers is an absolute necessity

1.4 Common Internet Protocols

The internet protocols are derived from the **TCP/IP** (Transmission Control Protocol/ Internet Protocol) suite of protocols which is the set of protocols used to communicate across the internet. It is also widely used on many organizational networks due to its flexibility and wide array of functionality provided. Microsoft who had originally developed their own set of protocols now is more widely using TCP/IP, at first for transport and now to support other services. For the purpose of this unit we will discuss the most common and important protocols.

ARP - Address Resolution Protocol enables the packaging of IP data into ethernet packages. It is the system and messaging protocol that is used to find the ethernet (hardware) address from a specific IP number. Without this protocol, the ethernet package could not be generated from the IP package, because the ethernet address could not be determined.

RARP - Reverse address resolution protocol is used to allow a computer without a local permanent data storage media to determine its IP address from its ethernet address.

IP- Internet Protocol, Except for ARP and RARP all protocols' data packets will be packaged into an IP data packet. IP provides the mechanism to use software to address and manage data packets being sent to computers.

TCP - A reliable connection oriented protocol used to control the management of application level services between computers. It is used for transport by some applications.

UDP - An unreliable connection less protocol used to control the management of application level services between computers. It is used for transport by some applications which must provide their own reliability.

FTP - File Transfer Protocol allows file transfer between two computers with login required.

HTTP - Hypertext Transfer Protocol is used to transport HTML pages from web servers to web browsers. The protocol used to communicate between web servers and web browser software clients.

BOOTP - the Bootstrap Protocol, or BOOTP, is a network protocol used by a network client to obtain an IP address from a configuration server. BOOTP is usually used during the bootstrap process when a computer is starting up. A BOOTP configuration server assigns an IP address to each client from a pool of addresses.

DHCP - Dynamic host configuration protocol is a method of assigning and controlling the IP addresses of computers on a given network. It is a server based service that automatically assigns IP numbers when a computer boots. This way the IP address of a computer does not need to be assigned manually. This makes changing networks easier to manage. DHCP can perform all the functions of BOOTP.

RIP - Routing Information Protocol is used to dynamically update router tables on WANs or the internet. A distance-vector algorithm is used to calculate the best route for a packet. RFC 1058, 1388 (RIP2).

OSPF - Open Shortest Path First dynamic routing protocol. A link state protocol rather than a distance vector protocol. It tests the status of its link to each of its neighbours and sends the acquired information to them.

POP3 - Post Office Protocol version 3 is used by clients to access an internet mail server to get mail. It is not a transport layer protocol.

IMAP4 - Internet Mail Access Protocol version 4 is the replacement for POP3.

Telnet is used to remotely open a session on another computer. It relies on TCP for transport and is defined by RFC854.

1.5 Internet Architecture

Internet is by definition a meta-network, a constantly changing collection of thousands of individual networks intercommunicating with a common protocol.

The Internet's architecture is described in its name, a short form of the compound word "inter-networking". This architecture is based in the very specification of the standard TCP/IP protocol, designed to connect any two networks which may be very different in internal hardware, software, and technical design. Once two networks are interconnected, communication with TCP/IP is enabled end-to-end, so that any node on the Internet has the near magical ability to communicate with any other no matter where they are. This openness of design has enabled the Internet to grow to a global scale.

The companies running the Internet backbone operate very high bandwidth networks relied on by governments, corporations, large organizations, and other Internet service providers. Their technical infrastructure often includes global connections through underwater cables and satellite links to enable communication between countries and continents.

Each communication packet goes up the hierarchy of Internet networks as far as necessary to get to its destination network where local Routing takes over to deliver it to the addressee. In the same way, each level in the hierarchy pays the next level for the bandwidth they use, and then the large backbone companies settle up with each other. Bandwidth is priced by large Internet service providers by several methods, such as at a fixed rate for constant availability of a certain number of megabits per second, or by a variety of use methods that amount to a cost per gigabyte. Due to economies of scale and efficiencies in management, bandwidth cost drops dramatically at the higher levels of the architecture.

The internet is based on client/server architecture where sever and the client resides in different machines. The **client–server model** (see Fig 1) is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server machine is a host that is running one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

Advantages client/server architecture

- In most cases, client/server architecture enables the roles and responsibilities of a computing system to be distributed among several independent computers that are known to each other only through a network. This creates an additional advantage to this architecture: greater ease of maintenance. For example, it is possible to replace, repair, upgrade, or even relocate a server while its clients remain both unaware and unaffected by that change.

- All data is stored on the servers, which generally have far greater security controls than most clients. Servers can better control access and resources, to guarantee that only those clients with the appropriate permissions may access and change data.
- Since data storage is centralized, updates to that data are far easier to administer in comparison to a P2P paradigm. In the latter, data updates may need to be distributed and applied to each peer in the network, which is time-consuming as there can be thousands or even millions of peers.
- Many mature client/server technologies are already available which were designed to ensure security, friendliness of the user interface, and ease of use.
- It functions with multiple different clients of different capabilities.

Disadvantages client/server architecture

- As the number of simultaneous client requests to a given server increases, the server can become overloaded. Contrast that to a P2P network, where its aggregated bandwidth actually increases as nodes are added, since the P2P network's overall bandwidth can be roughly computed as the sum of the bandwidths of every node in that network.
- The client/server paradigm lacks the robustness of a good P2P network. Under client/server, should a critical server fail, clients’ requests cannot be fulfilled. In P2P networks, resources are usually distributed among many nodes. Even if one or more nodes depart and abandon a downloading file, for example, the remaining nodes should still have the data needed to complete the download.

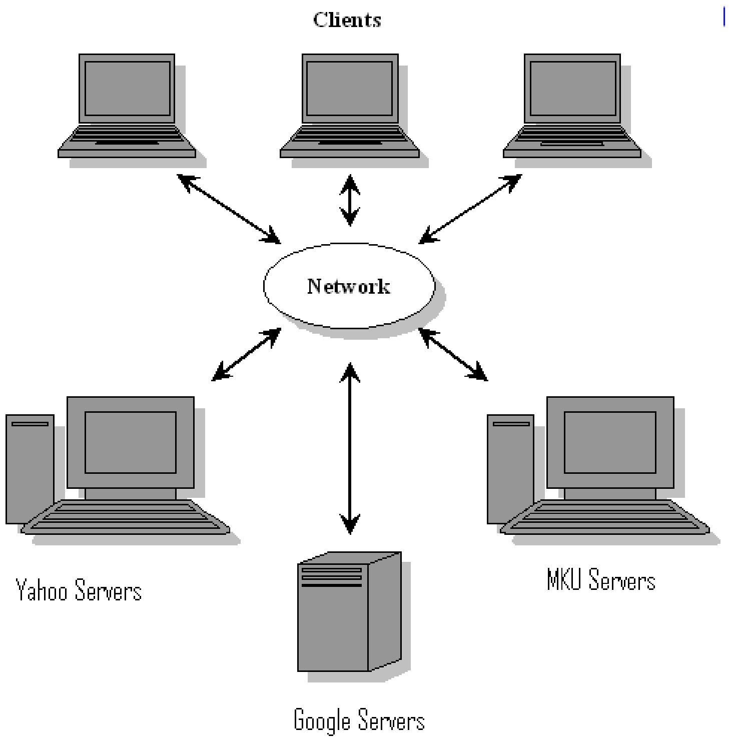


Fig 1.1: Client/Server Architecture

Chapter Review Questions

1. Explain in brief
    a. Web Conferencing
    b. Web crawler
    c. Top-level domain
    d. URLs
    e. Routers
    f. HTTP
    g. FTP
2. Explain in brief the concept of e-mail.
3. What are the basic objectives of FTP?
4. Outline the advantages of Client/Server architecture
5. Outline the disadvantages of Client/Server architecture

Suggested Further Reading

1. Niederst J., Learning Web design, Shroff Publishers

2. Peter, K., Web design, Tools and Techniques, Peach Pit press

3. http://www.w3schools.com

CHAPTER TWO: WEB PROJECT MANAGEMENT

Learning Objectives:

By the end of this chapter the learner shall be able to;

i. Explain the process of project management

ii. Explain online project management

iii. Estimate the cost and the size on an online project

2.1 Introduction to Project management

Project management is the discipline of planning, organizing, securing and managing resources to bring about the successful completion of specific project goals and objectives. A *project* is a temporary endeavor, having a defined beginning and end (usually constrained by date, funding and deliverables), undertaken to meet unique goals and objectives, usually to bring about beneficial change or additional value. The temporary nature of projects stands in contrast to business (or operations), which are repetitive, permanent or semi-permanent functional work to produce products or services.

The primary challenge of project management is to achieve all of the project goals and objectives while honoring the preconceived project constraints. Typical constraints are scope, time, and budget. The secondary, and more ambitious, challenge is to optimize the allocation and integration of inputs necessary to meet pre-defined objectives.

The following are the major project management processes/steps: -

1. *Initiation* - The initiation processes determine the nature and scope of the project. If this stage is not performed well, it is unlikely that the project will be successful in meeting the business needs. The initiation stage should include a plan that encompasses the following areas:

    - Analyzing the business needs/requirements in measurable goals
    - Reviewing of the current operations
    - Financial analysis of the costs and benefits including a budget
    - Stakeholder analysis, including users, and support personnel for the project

- Project charter including costs, tasks, deliverables, and schedule

2. *Planning and design* - After the initiation stage, the project is planned to an appropriate level of detail. The main purpose is to plan time, cost and resources adequately to estimate the work needed and to effectively manage risk during project execution. Project planning generally consists of: -

   - Determining how to plan
   - Developing the scope statement
   - selecting the planning team
   - identifying deliverables and creating the work breakdown structure
   - identifying the activities needed to complete those deliverables and networking the activities in their logical sequence
   - estimating the resource requirements for the activities
   - estimating time and cost for activities
   - developing the schedule
   - developing the budget
   - risk planning
   - gaining formal approval to begin work

3. *Executing* - Executing consists of the processes used to complete the work defined in the project management plan to accomplish the project's requirements. Execution process involves coordinating people and resources, as well as integrating and performing the activities of the project in accordance with the project management plan. The deliverables are produced as outputs from the processes performed as defined in the project management plan.

4. *Monitoring and controlling* - Monitoring and controlling consists of those processes performed to observe project execution so that potential problems can be identified in a timely manner and corrective action can be taken, when necessary, to control the execution of the project. The key benefit is that project performance is observed and measured regularly to identify variances from the project management plan. Monitoring and Controlling includes:

   - Measuring the ongoing project activities ('where we are')

- Monitoring the project variables (cost, effort, scope, etc.) against the project management plan and the project performance baseline (where we should be)
- Identify corrective actions to address issues and risks properly (How can we get on track again)
- Influencing the factors that could circumvent integrated change control so only approved changes are implemented

5. *Closing* - Closing includes the formal acceptance of the project and the ending. Administrative activities include the archiving of the files and documenting lessons learned. This phase consists of:
    - Project close: Finalize all activities across all of the process groups to formally close the project or a project phase
    - Contract closure: Complete and settle each contract (including the resolution of any open items) and close each contract applicable to the project or project phase.

2.2 Online project management systems

Online project management systems allows for project management tasks to be done over the Internet. Online project management systems allow any authorized person connected to the network to check the status of the project. The system centralizes all the information in a location that can be accessed easily by others.

Basic Online Project Management Functions

The purpose of an online project management system is to organize and simplify projects from start to finish. The following are the basic functions that almost every online project management system has:

- *Organizational systems* **-** keep track of files and documents. Most allow users to customize the filing system. The goal of the system is to make it easy to find any single piece of information in a project, no matter how complex that project may be.
- *Document management* - tracks document modifications, including the identity of the person who changed the file. Many systems archive older copies of files rather than delete them.
- *Scheduling* - is another key feature for most systems. Whether it's a centralized calendar with to-do lists or an automated schedule of e-mails to keep everyone informed and on task, scheduling plays a big role in project management. Many systems include streamlined

features that let managers keep track of who's doing what and the project's progress as a whole.

- *Time tracking* **-** with this function, software keeps track of how much time people spend working on each part of a project. Many companies bill customers based on how much time was spent on a particular project. Others might pay employees at an hourly rate. Time tracking applications let the company keep tabs on how many hours each person in the project has worked on various phases of the project.

- *Documentation* – this involves keeping a detailed history of projects, which can help protect a company from liability. For example, let's say a construction company gets a request to build a house. The client makes a change to the house's design early in the project. The construction company makes a note of the client's requests and documents it using an online project management system. Later, when construction is complete, the client expresses dissatisfaction with the way the house looks due to the changes in design. With the documentation feature, the construction company has a record that can help it prove that the changes were at the client's request.

- *Issue tracking* - lets users create a file to report any problems or issues that crop up during a project. These applications have a feature that e-mails the appropriate parties so that the issue can be resolved as quickly as possible. Once someone fixes the problem, he or she can note it in the issue report. The system then files the report as part of the documentation function.

- *Communication systems* - might include a message board, instant messaging program, chat room or some other communication forum. Including a message system within the overall system makes it easier for people involved in the project to pass information along.

2.3 Examples of web projects

- *New web side design and development.*

- *Web Site Redesign:* if it has been more than 3 years since your last design, it is definitely time to at least take a serious look at your web site

- *Adding Functionalities to Your Web Site:* time to start using ecommerce to sell products or process donations, add an interactive calendar, add social net working, etc.

- *Search Engine Optimization:* if you’d like to capture more search traffic for your products or services, it simply won’t happen without conscious SEO

- *Web Site or Internet Marketing*: if you build it, they will come is not a sound business practice, you need a plan to promote awareness of your site
- *Email Campaigns:* email has enjoyed a revival and when a campaign is properly constructed, the results can be quite extraordinary
- *Developing a mobile version.*

2.4 Choosing a Web development firm

In order to find a Web design firm best suited for your project, your organization should be aware of the following:

- The intended goal of the project
- The intended audience for the Web site
- The anticipated budget for the Web site development
- How the project will fit into a larger marketing/corporate scheme
- The individuals/departments within your organization responsible for the project

Five steps to follow when choosing a Web design company.

- Self Analysis - define your organization's requirements. That said, the better you can define your Web development and marketing goals the better it is – for both you and the potential Web development firm.
- Generate a list – gather a list of potential Web design from referrals, analysis of other sites you admire and from list provided by a consultant.
- Create a shortlist - From the list of potential Web design firms, your organization's goal should now be to create a shortlist of 3-5 firms that are best suited for your project. In order to determine which Web design firms are most appropriate for your project, do the following:
    - Examine their portfolio
    - Technical competence and experience
    - Analyze the process
    - Determine what other products/services the Web design company can offer

- Get proposals - Once you have selected your short-list of top Web development firms, request that they send you a proposal. Ask that their proposal include an overview of your requirements and their proposed solution. This will help to determine which Web design firm understands your requirements best. Also ask that each Web design firm include a description of their development process and a price breakdown for the various aspects of the Web site project.

- Evaluate and select a partner - Start off by examining each proposal individually. After your organization has reviewed all proposals, compare them with each other. Once you have made a decision, contact the top Web development firm and let them know the good news.

2.5 Budget Planning for the Web site development project

First, make sure the scope is defined for the Web presence. A documented Web operations strategy will help you define budget parameters so when you make a request, you can clearly state that you're seeking funding for "X" Web property.

Next, state the goals for the Web property and the planned strategies and tactics for meeting those goals. This process will allow you to identify quantifiable success metrics and provide rationale for funding.

At this point, you can begin to assign costs to each line item. Make sure you take into consideration the costs for hardware, software, design, implementation and maintenance. Other cost considerations are training, communications and change management.

Web goals should include specific projects (i.e. a site re-design) as well as day-to-day operations. Your budget should account for all the things that are required for maintaining a quality Web presence. This should include daily upkeep of design and editorial content, ongoing usability and analytics, R&D, and maintenance of Web tools and infrastructure.

Depending on how your organization budgets, you may have a stand-alone Web budget or you may need to collaborate with other departments to make sure they're accounting for the Web project in their budgets.

A detailed Web plan for upcoming projects and day-to-day operations will help everyone understand priorities and costs for the Web presence. Share the plan with your Web stakeholders to ensure buy-in and support for the budget.

The following are the main reasons why web projects get over budget: -

Frequently changing requirements - rigid requirements make it difficult to react to the changes that inevitably occur as knowledge and environments evolve. Requirements that have been omitted are generally picked up late in the process, by which time they are awkward and costly to implement.

- Too many stakeholders having a say in the web development process
- Not enough budget or time being allocated

How to ensure that web design and development projects are delivered in budget: -

- Take a collaborative approach to Web project development - The approach would involve regular meetings with all stakeholders, where working software is tested and enough quality assurance is carried out.
- By using a combination of web development, project management and software engineering tools
- Making sure that requirements were collected exhaustively
- Allocating enough time and budget to the web project

2.6 Cost estimation for web projects

Software cost models and effort estimates help project managers allocate resources, control costs and schedule and improve current practices, leading to projects finished on time and within budget. In the context of Web development, these issues are also crucial and very challenging given that Web projects have short schedules and very fluidic scope. Web projects cost estimation is the process through which the web developer decides how much the development process of the website will cost the organization.

For Web development, cost is difficult to estimate because:

- There is no standard to sizing Web applications. Each can be created using diverse technologies such as several varieties of Java (Java, servlets, Enterprise java Beans, applets, and Java Server Pages), HTML, JavaScript, XML, XSL, and so on.
- Web project's primary goal is to bring quality applications to market as quickly as possible, varying from a few weeks
- People involved in Web development are represented by less experienced programmers, users as developers, graphic designers and new hires straight from university

- Processes employed are in general heuristic, although some organizations are starting to look into the use of agile methods
- Web projects have short schedules and a fluidic scope
- Typical project size is small, using 3 to 7 team members

Several techniques for cost and effort estimation have been proposed over the last 30 years in software engineering, falling into two general categories: -

1. *Expert judgement (EJ)* – EJ has been widely used. However, the means of deriving an estimate are not explicit and therefore not repeatable. Expert opinion, although always difficult to quantify, can be an effective estimating tool on its own or as an adjusting factor for algorithmic models
2. *Algorithmic models (AM)* –attempts to represent the relationship between effort and one or more project characteristics. The main "cost driver" used in such a model is usually taken to be some notion of software size (e.g. the number of lines of source code, number of pages, and number of links). Algorithmic models need calibration or adjustment to local circumstances.

2.7 How to measure the size of the website

Cowderoy (1998; 2000) recommends several size measures for cost estimation and risk assessment of Web application development projects. Measures have been organised into the following categories: -

- ***Web-site design.*** Creation and organisation of Web pages. **Length measures** provided are Web pages (number of Web pages), home pages (number of major entry points to the Web application/site), leaf nodes (number of leaf Web pages, i.e., pages that have no siblings), hidden nodes (number of pages excluded from the main navigation buttons), depth (number of nodes on the second level that have siblings); **complexity measures** are interconnectivity (number of URLs that link to other pages in the same application/site), external hyperlinks (number of unique URLs in the Web application/site); **functionality measures** are actions (number of independent actions provided from a Web page by use of Javascript, Active X and Java applets).
- **Technical authoring.** Collection of text and related hyperlinks, and text structuring sections etc. **Length measures** proposed are paragraph count (number of paragraphs in a Web page, Web application, or external document, i.e., PDF document), word count

(number of words in a Web page or document); **complexity measures** are navigational structures (number of different structures in a Web page or document).

- ***Data entry.*** Creation and maintenance of databases used by a Web application. **Length measures** proposed are key record types (number of different tables accessed by the application/site), key entries (number of keys in a database table), entry fields (number of table attributes manipulated by the application/site), secure fields (number of fields that contain safety-critical or financially secure information).
- ***Graphics design.*** Creation of 2D images using graphics design tools. **Length measures** are delivered images (number of unique images used by the Web application/site), image size (width * height), image composites (number of layers from which the final image was created), language versions (the number of versions of an image that must be produced to accommodate different languages or different cultural priorities).
- ***3D and VR(Virtual Reality) design.*** Creation of 3D objects and environments with which the user can interact. **Length measures** are 3d objects (number of files including one or more 3D objects used in the Web application/site) and virtual worlds (number of files including one or more virtual worlds used in the Web application/site).
- ***Sound engineering.*** Adaptation of existing audio tracks for use at Web applications. **Length measures** are audio files (number of unique audio files used in the Web application/site), duration (summed duration of all sequences within an audio file), audio sequences (number of sequences within the audio file), tracks (number of layers within an audio sequence), audio edits (number of edits applied to groups of frames within a sequence, e.g., fades) and imported images (number of separate graphics images imported into an interactive audio file).
- ***Animation and video editing.*** Building movies from existing images, video, text and sound. **Length measures** are movies (number of movie files used in a Web application, Web page, or referenced by other movies at this Web application or Web page), duration of a movie, and story composites (for each scene, count the number of layers from which the final image was created), frame rate (number of frames per second as defined in the delivered file).
- ***Programming.*** Creation of server-side software and client-side programs that replace browsers. Classifies measures into three categories – **Specification, Design and Program**. The **Specification length measures** proposed are user commands (number of different menu commands, buttons, etc that the user can access from the application's interface), database files (number of entities and relationships in the database schema) and classes definitions (object-oriented classes resulting from the analysis of use cases). The **Design length measures** are any design object-oriented measures where no one in

particular is specified. The **Design functionality measures** are Function Points or Full Function Points. Finally, **Program length measures** are lines of source code.

Chapter Review Questions

1. Define project management.
2. Outline the major steps of project management
3. What are the major functions of online project management
4. What are the main reasons why web projects are over budget?

References

1. Cowderoy, A.J.C., Donaldson, A.J.M., Jenkins, J.O. 1998. A Metrics framework for multimedia creation, Proc. 5th IEEE International Software Metrics Symposium, Maryland, USA.
2. Cowderoy, A.J.C., 2000. Measures of size and complexity for web-site content, Proc. Combined 11th ESCOM Conference and the 3rd SCOPE conference on Software Product Quality, Munich, Germany.

Suggested Further Reading

1. Niederst J., Learning Web design, Shroff Publishers
2. Peter, K., Web design, Tools and Techniques, Peach Pit press
3. http://www.w3schools.com

CHAPTER THREE: WEBSITE INTERFACE DESIGN

Learning Objectives:

By the end of this chapter the learner shall be able to;

i. design a web page
ii. explain the website design basics
iii. explain the principle of user interface design
iv. explain the key elements of a good website

3.1 Introduction Web Interface Design

Website interface design involves making decisions on the look and feel of the website. This starts when the web developer collects the site requirements from the users. Using the defined user needs the website developer is able to decide which design is most appropriate for a certain website. User interface is the space where interaction between humans and website occurs. The goal of interaction between a human and a website at the user interface is effective operation and control of the website and feedback from the website which aids the user in make certain decisions. User interfaces provide a means of:

- Input, allowing the users to manipulate a website
- Output, allowing the website to indicate the effects of the users' manipulation

Web design is the process of designing websites - a collection of online content including documents and applications that reside on web servers. The process of web design includes planning, post-production, research, advertising, as well as media control that is applied to the pages within the site by the designer or group of designers with a specific purpose. The site itself can be divided into its main page, also known as the home page, which cites the main objective as well as highlights of the site's daily updates; which also contains hyperlinks that functions to direct viewers to a designated page within the site's domain.

Website Design Basics

Following are some basic steps to be followed while creating web pages:

- Create a Welcom/Index/first page of your web site.
- Avoid use of heavy graphics in the first page from the user‘s point of view.
- Page should be descriptive and interactive both as per requirements.

Some minimum technical requirements might include:

- Fast loading of pages
- Presentation with clarity and readability - with or without graphics
- Validation of the contents
- Easy and clear navigation
- Instruction on how to use the site.

Identify and address all potential problems such as:

- Slow connections
- Physical constraints of the user such as sight or hearing impaired visitors
- Platform support
- Browser support

Web page guidelines: Design Considerations

- Follow a simple and consistent design. Complex designs can confuse people, so keep it simple. A consistent design will let your readers concentrate on content, without having to waste time figuring out how to maneuver your layout. At Cornell, the time and workstation "power" required to load a Web page are still important considerations when using some of the latest Web authoring tools, especially if your visitor is connecting to campus with a modem.
- Don't create gratuitous graphics. Graphics are one reason for the interest in the Web by both publishers and readers and should definitely be included on your Web page if possible. Still, it's important not to overuse them. Blinking text and other excessive decorations can be distracting; background colors and textures can affect download time. Think very carefully about the colors and textures you choose for backgrounds and the effects they'll have on the readability of text. Also, don't forget that many people still use monochrome monitors.

- Give people cross links. Visitors should be able to move from one major page to another on your site without having to go back to your home page. Put cross links to all your major pages at the bottom of all major pages. For a good example, see the bottom of this page.
- Be careful about "over-linking." While linking to the work of others in your organization and throughout the world can help your readers, it can also lead to information overload. It's important to balance linking within your page design. Too many links can be a visual eyesore on the page and a distraction from the original information you want your readers to concentrate on.
- Don't create dead end links. Readers can get discouraged from returning to your pages when those pages are filled with empty links with grand labels like Descriptions of All Classes!

3.2 Principles of User Interface Design

Principles of User Interface Design are intended to improve the quality of user interface design:

- ***The structure principle***: Design should organize the user interface purposefully, in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another. The structure principle is concerned with overall user interface architecture.
- ***The simplicity principle***: The design should make simple, common tasks easy, communicating clearly and simply in the user's own language, and providing good shortcuts that are meaningfully related to longer procedures.
- ***The visibility principle***: The design should make all needed options and materials for a given task visible without distracting the user with extraneous or redundant information. Good designs don't overwhelm users with alternatives or confuse with unneeded information.
- ***The feedback principle***: The design should keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.
- ***The tolerance principle***: The design should be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing, while also preventing errors wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions.

- ***The reuse principle***: The design should reuse internal and external components and behaviors, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember.

3.3 Web usability

Web usability is an approach to make web sites easy to use for an end-user, without the requirement that any specialized training be undertaken. The user should be able to intuitively relate the actions he needs to perform on the web page, with other interactions he sees in the general domain of life e.g. press of a button leads to some action. The broad goal of usability can be:

- Present the information to the user in a clear and concise way.
- To give the correct choices to the users, in a very obvious way.
- To remove any ambiguity regarding the consequences of an action e.g. clicking on delete/remove/purchase.
- Put the most important thing in the right place on a web page or a web application.

3.4 Key Elements of a Good Website

The objective of any website is to get viewed and read as much as possible. Elements such as content, appearance and usability are some of the factors that set apart a great website from a good one. There are several factors that go into making a successful website. Besides its use, the appearance and user-friendliness are the most important factors that help make a website popular. Following are the key elements of a good website:

- Content - The most important element in any website is good content. Your content is the reason users visit your site. If it is superficial or badly written, you can be sure that they will leave fast and won't come back again. The copy for your website should be well framed and helpful Truthful information about your company, products, services, etc is a must. Information one must not forget to include is an 'about' page as well as contact information. The important aspect to keep in mind here is to provide information that is going to be useful to your user. Also, ensure that the content on your website is updated and does not contain any grammatical or spelling mistakes, for it is a direct reflection on your company's image.
- Visual Appeal - We all know what they say about first impressions, and this school of though extends to websites too. The decision to go through the content of a website is likely to be based on how a visitor responds to its appearance. Your website design should be clean and simple, but all the same - eye-catching. A tasteful website isn't flashy. Blinking letters and

flashing pictures are disturbing, and will turn away most readers instead of attracting them. They also make it a lot harder to read the information provided, which can be quite irritating for the user.

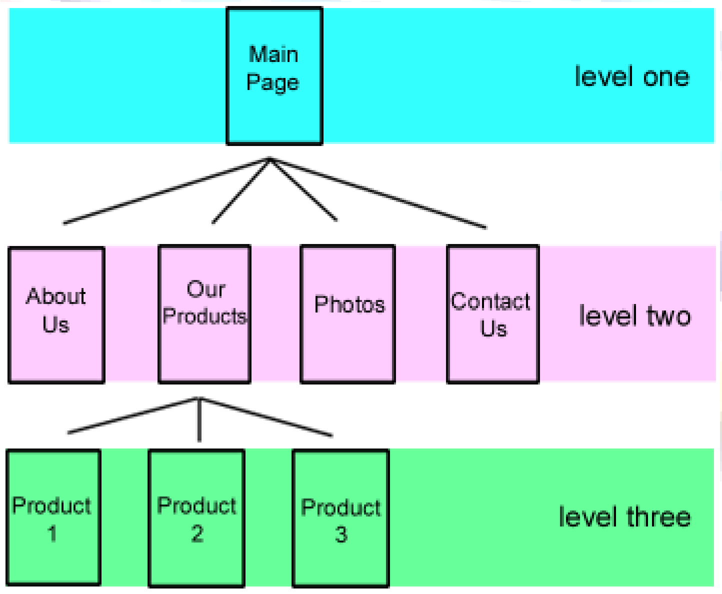
- User Interface - A good user interface is another key element of a good website. The user interface should be the kind an average user will find easy to use, and enable him or her to find whatever they want with ease. Part of this includes well-planned major (top level) categories, navigation that is easy to spot and consistent throughout the website. One way to achieve this is to have links to the key areas of your site in the top half of your website. This layout should remain consistent throughout the various pages. A good user interface will offer the same options in the footer, or even a few extra links. Larger websites with lots of pages should have a site map.
- Content Layout - A big mistake is to place text over backgrounds, for it makes reading difficult. Dark background colors do not make reading any easier. Studies have shown that black text on a white background is the best combination for reading and remembering. Large blocks of text should be split up into paragraphs containing 4-5 sentences. Ensure that your page contains sufficient white space, for clarity and neatness. Every page should have the same margins. There should be consistency throughout the website, so only one set of fonts, colors and layout should be used on all pages. On element that annoys most users is multiple scroll bars. Keep them at a minimum and try to avoid the side-to-side scroll bars.
- Search Tool - Good websites containing many pages have a search field to assist their readers. This feature is very helpful, and enables potential customers to find whatever they want with ease. One can make use of use a Google Search on their site, or WordPress (or another blogging platform or CMS / Content Management System). It will be a little more challenging with a static html site, but there are services available that will let one incorporate a functional search box onto their site.
- Separate Design from Content - The best developed sites on the Internet combine the use of XHTML and CSS (Cascading Style Sheets), which create a separation of design vs content. Among its many advantages are that one has to update just one CSS file to see the change implemented site-wide. Also this separation of content from design will result in quick loading (search engines will no longer have to go through excess code to find out if the content is relevant). On the same note, minimize your utilization of flash animations, music, or video or anything that is going to make your site take a long time to load. Web surfers have very little patience and a short attention span.

- Cross Browser Compatibility - Another key element of a good website is that it displays well in as many of the mainstream web browsers as possible and is also compatible across platforms. Although most Internet users utilize Firefox, you don't want to leave out those using Internet Explorer and the other few browsers. Also, the success of your website can be hinged on how well it is optimized for search engines.

- Web Optimized Images - Good picture quality on a website instantly leaves a user impressed. Most people save all their images in a compressed format. So be careful while doing this as your images will appear pixelated when you upload them on your site.

3.5 Web Site Organization

A well organized web site will increase its usability resulting in your visitors staying on your site longer and coming back more frequently. The main page of your site, also referred to as the home or index page, is the first page visitors will see when going to your domain. On this page there should be navigation linking to each of the major subpages of your site. These navigation links can be either text links or image buttons. For example, imagine a fictional company who wants a web site as a sort of online brochure. They have 3 products to sell, they don't want to sell them online, but would like detailed information about each of them to be available on their site. They also want an About Us, a Contact Us, and a Photos page. A products page could be set up that gives an introduction about their products, and then links to 3 separate pages each of which describes one of the products in detail. The main page could be referred to as level one, the Our Products, About Us, Contact Us, and the Photos page as level 2, and each of the individual product pages as level three. See diagram below.



There would then be five main navigational links for this site:

- Main page
- About Us
- Our Products
- Photos
- Contact Us

These five links should appear on each of the eight pages of the site. The header and title on the top of each page should either be the same as or similar to the text on each of the navigation links. In this way whichever page of the site a visitor finds themselves on, either through moving around the site on their own, or by entering the site on any of its eight pages through a search engine link, they should be able to easily figure out exactly where on the site they are.

To maintain a consistent look and feel throughout the site, the same overall design should be used on each page. For example, if each of the five main navigation links were made from buttons shaped like hats, and they were lined up along a left column, they should be that way on each page of the site. It is however, a good idea to include plain text links to each of the main pages on the bottom of each page of the site. This is especially important if the other links are made from graphics. There are people who use browsers with either no graphics capability, or they have them turned off. They will need a way to navigate through your site also.

It has also been stated that it should not take more than three clicks to get from one page to any other page on the same web site, and if it does, the site just isn't organized well.

3.6 Website wireframe

A website wireframe, also known as a page schematic or screen blueprint, is a visual guide that represents the skeletal framework of a website. The wireframe depicts the page layout or arrangement of the website’s content, including interface elements and navigational systems, and how they work together. The wireframe usually lacks typographic style, color, or graphics, since the main focus lies in functionality, behavior, and priority of content. Website wireframe focuses on “what a screen does, not what it looks like.” Wireframes focus on:

- The kinds of information displayed
- The range of functions available
- The relative priorities of the information and functions

- The rules for displaying certain kinds of information
- The effect of different scenarios on the display

The website wireframe connects the underlying conceptual structure, or information architecture, to the surface, or visual design of the website. Wireframes help establish functionality, and the relationships between different screen templates of a website. An iterative process, creating wireframes is an effective way to make rapid prototypes of pages, while measuring the practicality of a design concept. Wireframing typically begins between “high-level structural work—like flowcharts or site maps—and screen designs. Within the process of building a website, wireframing is where thinking becomes tangible.

Uses of wireframes

Developers use wireframes to get a more tangible grasp of the site’s functionality, while designers use them to push the user interface (UI) process. User experience designers and information architects use wireframes to show navigation paths between pages. Business stakeholders use wireframes to ensure that requirements and objectives are met through the design.

Working with wireframes may be a collaborative effort since it bridges the information architecture to the visual design. Due to overlaps in these professional roles, conflicts may occur, making wireframing a controversial part of the design process. Since wireframes signify a “bare bones” aesthetic, it is difficult for designers to assess how closely the wireframe needs to depict actual screen layouts. Another difficulty with wireframes is that they don’t effectively display interactive details. Modern UI design incorporates various devices such as expanding panels, hover effects, and carousels that pose a challenge for 2-D diagrams. Wireframes may have multiple levels of detail and can be broken up into two categories in terms of fidelity, or how closely they resemble the end product.

- Low-fidelity Resembling a rough sketch or a quick mock-up, low-fidelity wireframes have less detail and are quick to produce. These wireframes help a project team collaborate more effectively since they are more abstract, using rectangles and labeling to represent content.
- High-fidelity High-fidelity wireframes are often used for documenting because they incorporate a level of detail that more closely matches the design of the actual webpage, thus taking longer to create.

Elements of wireframes

The skeleton plan of a website can be broken down into three components: information design, navigation design, and interface design. Page layout is where these components come together, while wireframing is what depicts the relationship between these components.

- Information Design - Information design is the presentation—placement and prioritization of information in a way that facilitates understanding. Information design is an area of graphic design, meant to display information effectively for clear communication. For websites, information elements should be arranged in a way that reflects the goals and tasks of the user.

- Navigation Design - The navigation system provides a set of screen elements that allow the user to move page to page through a website. The navigation design should communicate the relationship between the links it contains so that users understand the options they have for navigating the site. Often, websites contain multiple navigation systems such as a global navigation, local navigation, supplementary navigation, contextual navigation, and courtesy navigation.

- Interface Design - User interface design includes selecting and arranging interface elements to enable users to interact with the functionality of the system. The goal is to facilitate usability and efficiency as much as possible. Common elements found in interface design are action buttons, text fields, check boxes, radio buttons and drop-down menus.

Chapter Review Questions

1. Can you give the minimum technical requirements that you can outline for a new site
2. outline the principles of user interface design
3. outline the key elements of a good website
4. define a website wireframe
5. what are the elements of a wireframe

Suggested Further Reading

1. Niederst J., Learning Web design, Shroff Publishers

2. Peter, K., Web design, Tools and Techniques, Peach Pit press

3. http://www.w3schools.com

CHAPTER FOUR: HTML PAGE LAYOUT DESIGN

Learning Objectives:

By the end of this chapter the learner shall be able to;

i. Create HTML forms

ii. Create HTML iFrames

iii. Create HTML layouts using tables and Div elements

1.1 HTML Forms

A form on a web page allows a user to enter data that is sent to a server for processing. forms resemble paper or database forms because internet users fill out the forms using checkboxes, radio buttons, or text fields. For example, forms can be used to enter shipping or credit card data to order a product or can be used to retrieve data (e.g., searching on a search engine).

In addition to functioning as input templates for new information, forms can also be used to query and display existing data in a similar manner to mail merge forms, with the same advantages. The decoupling of message structure and underlying data allow both to vary independently. The use of forms for this purpose avoids the problems associated with explicitly creating separate web pages for each record in a database.

HTML forms are used to pass data to a server. A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements. The <form> tag is used to create an HTML form:

```
<form>
.
input elements
.
</form>
```

The Input Element

The most important form element is the input element. The input element is used to select user information. An input element can vary in many ways, depending on the type attribute. An input element can be of type text field, checkbox, password, radio button, submit button, and more. The most used input types are described below.

1. *Text Fields*

<input type="text" /> defines a one-line input field that a user can enter text into:

```
<form>
First name: <input type="text" name="firstname" /><br />
Last name: <input type="text" name="lastname" />
</form>
```

How the HTML code above looks in a browser:

First name:

Last name:

Note: The form itself is not visible. Also note that the default width of a text field is 20 characters.

2. Password Field

<input type="password" /> defines a password field:

```
<form>

Password: <input type="password" name="pwd" />

</form>
```

How the HTML code above looks in a browser:

Password: ****

Note: The characters in a password field are masked (shown as asterisks or circles).

3. Radio Buttons

<input type="radio" /> defines a radio button. Radio buttons let a user select ONLY ONE one of a limited number of choices:

```
<form>

<input type="radio" name="sex" value="male" /> Male<br />

<input type="radio" name="sex" value="female" /> Female

</form>
```

How the HTML code above looks in a browser:

○ Male

○ Female

4. *Checkboxes*

<input type="checkbox" /> defines a checkbox. Checkboxes let a user select ONE or MORE options of a limited number of choices.

```
<form>

<input type="checkbox" name="vehicle" value="Bike" /> I have a bike<br />

<input type="checkbox" name="vehicle" value="Car" /> I have a car
```

</form>

How the HTML code above looks in a browser:

☐ I have a bike

☐ I have a car

5. Submit Button

<input type="submit" /> defines a submit button. A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

<form name="input" action="html_form_action.asp" method="get">

Username: <input type="text" name="user" />

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

</form>

How the HTML code above looks in a browser:

Username: ☐

If you type some characters in the text field above, and click the "Submit" button, the browser will send your input to a page called "html_form_action.asp". The page will show you the received input.

1.2 HTML Iframes

An iframe is used to display a web page within a web page.

Syntax for adding an iframe:

<iframe src="URL"></iframe>

The URL points to the location of the separate page.

Iframe - Set Height and Width

The height and width attributes are used to specify the height and width of the iframe. The attribute values are specified in pixels by default, but they can also be in percent (like "80%").

Example

<iframe src="demo_iframe.htm" width="200" height="200"></iframe>

Iframe - Remove the Border

The frameborder attribute specifies whether or not to display a border around the iframe. Set the attribute value to "0" to remove the border:

Example

<iframe src="demo_iframe.htm" frameborder="0"></iframe>

Use iframe as a Target for a Link

An iframe can be used as the target frame for a link. The target attribute of a link must refer to the name attribute of the iframe:

Example

<iframe src="demo_iframe.htm" name="iframe_a"></iframe>

<p><a href="http://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>

1.3 HTML Layouts

Web page layout is very important to make your website look good. Design your webpage layout very carefully. Most websites have put their content in multiple columns (formatted like a magazine or newspaper). Multiple columns is created by using <table> or <div> tags. Some CSS are normally also added to position elements, or to create backgrounds or colorful look for the pages.

1. **HTML Layouts - Using Tables**

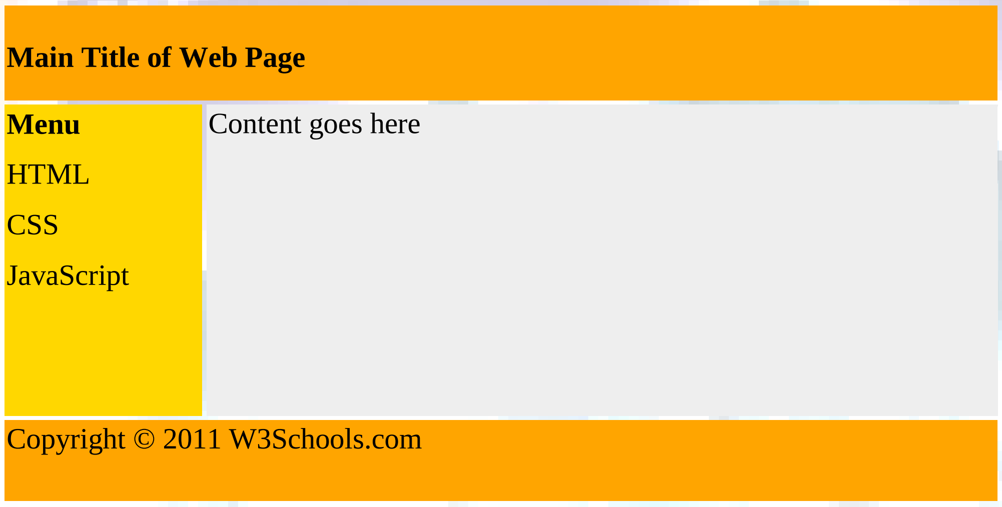
The simplest way of creating layouts is by using the HTML <table> tag. The following example uses a table with 3 rows and 2 columns - the first and last row spans both columns using the colspan attribute:

Example

```
<html>

<body>

<table width="500" border="0">

<tr>

<td colspan="2" style="background-color:#FFA500;">

<h1>Main Title of Web Page</h1>

</td>

</tr>

<tr valign="top">

<td style="background-color:#FFD700;width:100px;text-align:top;">

<b>Menu</b><br />

HTML<br />

CSS<br />

JavaScript

</td>

<td style="background-color:#EEEEEE;height:200px;width:400px;text-align:top;">

Content goes here</td>

</tr>

<tr>

<td colspan="2" style="background-color:#FFA500;text-align:center;">
```

```
Copyright © 2011 W3Schools.com</td>

</tr>

</table>

</body>

</html>
```

The HTML code above will produce the following result:



Main Title of Web Page

Menu
HTML
CSS
JavaScript

Content goes here

Copyright © 2011 W3Schools.com

Note: Even though it is possible to create nice layouts with HTML tables, tables were designed for presenting tabular data - NOT as a layout tool

2. HTML Layouts - Using Div Elements

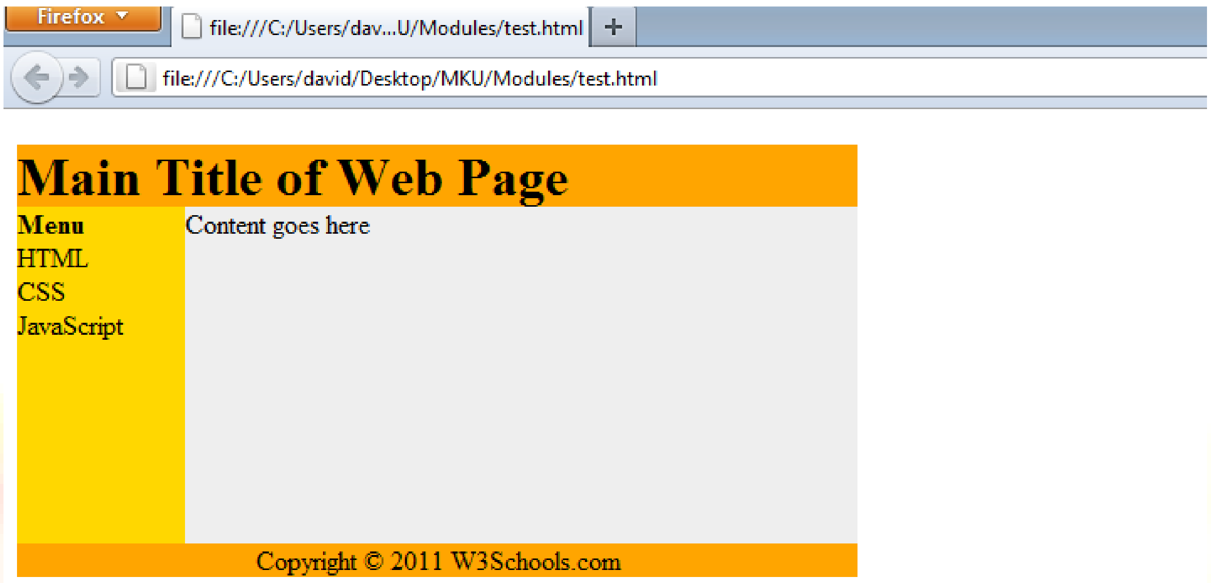
The div element is a block level element used for grouping HTML elements. The following example uses five div elements to create a multiple column layout, creating the same result as in the previous example:

Example

```
<html>

<body>

<div id="container" style="width:500px">
```

```
<div id="header" style="background-color:#FFA500;">

<h1 style="margin-bottom:0;">Main Title of Web Page</h1></div>

<div id="menu" style="background-color:#FFD700;height:200px;width:100px;float:left;">

<b>Menu</b><br />

HTML<br />

CSS<br />

JavaScript</div>

<div id="content" style="background-color:#EEEEEE;height:200px;width:400px;float:left;">

Content goes here</div>

<div id="footer" style="background-color:#FFA500;clear:both;text-align:center;">

Copyright © 2011 W3Schools.com</div>

</div>

</body>

</html>
```

The HTML code above will produce the following result:


Firefox
file:///C:/Users/dav...U/Modules/test.html
file:///C:/Users/david/Desktop/MKU/Modules/test.html

Main Title of Web Page

Menu
HTML
CSS
JavaScript

Content goes here

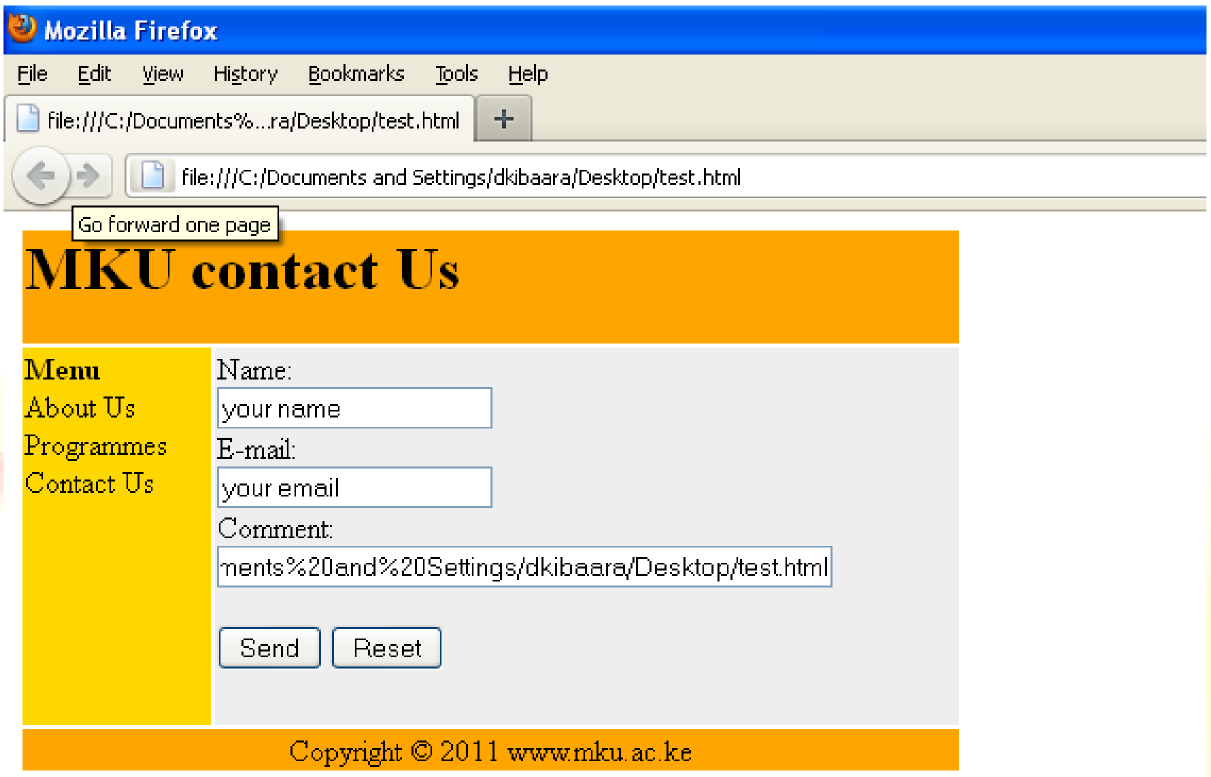
Copyright © 2011 W3Schools.com

HTML Layout - Useful Tips

The biggest advantage of using CSS is that, if you place the CSS code in an external style sheet, your site becomes MUCH EASIER to maintain. You can change the layout of all your pages by editing one file. Because advanced layouts take time to create, a quicker option is to use a template.

Chapter Review Questions

1. Define a HTML form
2. Write the HTML code to display the following window


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Documents%...ra/Desktop/test.html
file:///C:/Documents and Settings/dkibaara/Desktop/test.html
Go forward one page

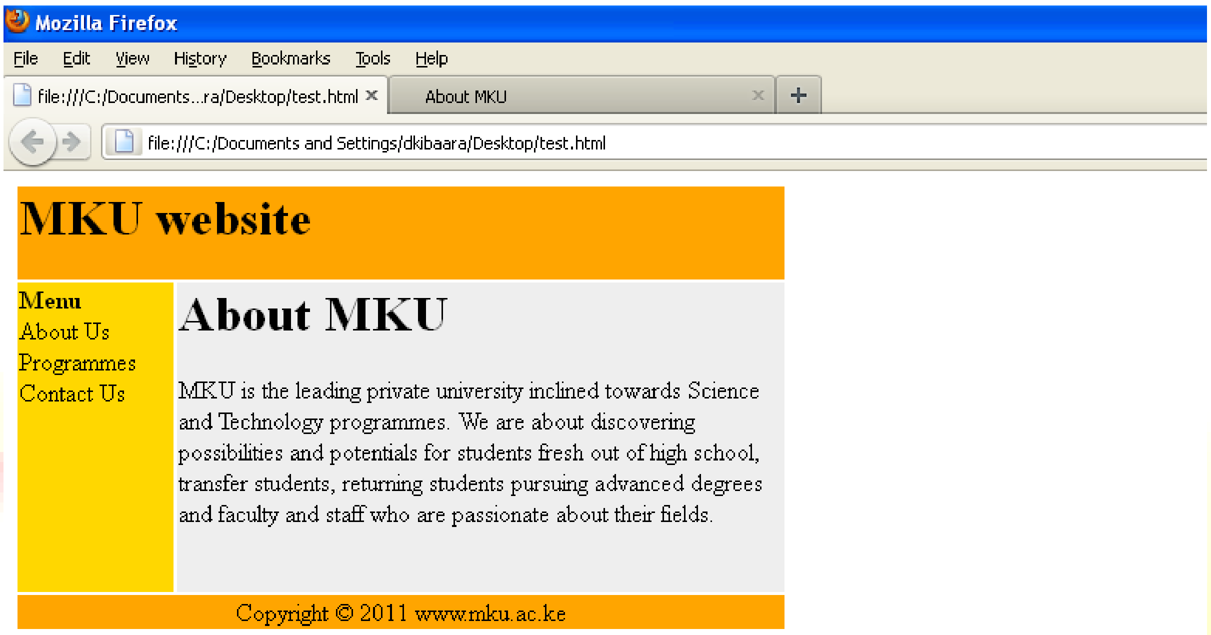
MKU contact Us

Menu
About Us
Programmes
Contact Us

Name:
your name
E-mail:
your email
Comment:
ments%20and%20Settings/dkibaara/Desktop/test.html

Copyright © 2011 www.mku.ac.ke

3. Using a table write the HTML code to display the following window


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Documents...ra/Desktop/test.html
About MKU
file:///C:/Documents and Settings/dkibaara/Desktop/test.html

MKU website

Menu
About Us
Programmes
Contact Us

About MKU

MKU is the leading private university inclined towards Science and Technology programmes. We are about discovering possibilities and potentials for students fresh out of high school, transfer students, returning students pursuing advanced degrees and faculty and staff who are passionate about their fields.

Copyright © 2011 www.mku.ac.ke

Suggested Further Reading

1. Niederst J., Learning Web design, Shroff Publishers

2. Peter, K., Web design, Tools and Techniques, Peach Pit press

3. http://www.w3schools.com

CHAPTER FIVE: PAGE LAYOUT: INTRODUCTION TO CSS (CASCADING STYLE SHEETS)

Learning Objectives:

By the end of this chapter the learner shall be able to;

i. Explain what is CSS

ii. Create CSS document and format WebPages using CSS

iii. Create CSS box model

5.1 Introduction to CSS

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation semantics (the look and formatting) of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document, including plain XML, SVG and XUL.

CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content. CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. While the author of a document typically links that document to a CSS style sheet, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified.

CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities or weights are calculated and assigned to rules, so that the results are predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318.

CSS Variations

CSS has various levels and profiles. Each level of CSS builds upon the last, typically adding new features and typically denoted as CSS 1, CSS 2, and CSS 3. Profiles are typically a subset of one or more levels of CSS built for a particular device or user interface. Currently there are profiles for mobile devices, printers, and television sets.

1. *CSS 1*

The first CSS specification to become an official W3C Recommendation is CSS level 1, published in December 1996. Among its capabilities are support for:

- Font properties such as typeface and emphasis
- Color of text, backgrounds, and other elements
- Text attributes such as spacing between words, letters, and lines of text
- Alignment of text, images, tables and other elements
- Margin, border, padding, and positioning for most elements
- Unique identification and generic classification of groups of attributes

2. *CSS 2*

CSS level 2 specification was developed by the W3C and published as a Recommendation in May 1998. A superset of CSS 1, CSS 2 includes a number of new capabilities like absolute, relative, and fixed positioning of elements and z-index, the concept of media types, support for aural style sheets and bidirectional text, and new font properties such as shadows.

CSS level 2 revision 1 or CSS 2.1 fixes errors in CSS 2, removes poorly-supported or not fully interoperable features and adds already-implemented browser extensions to the specification. In order to comply with the W3C Process for standardizing technical specifications, CSS 2.1 went back and forth between Working Draft status and Candidate Recommendation status for many years. CSS 2.1 first became a Candidate Recommendation on February 25, 2004, but it was reverted to a Working Draft on June 13, 2005 for further review. It was returned to Candidate Recommendation status on 19 July 2007 and was updated twice in 2009. However, since changes and clarifications were made to the prose it went back to Last Call Working Draft on 7 December 2010. Later it went into Proposed Recommendation on 12 April 2011. It was published as a Recommendation on 7 June 2011.

3. *CSS 3*

Instead of defining all features in a single, large specification like CSS 2, CSS 3 is divided into several separate documents called "modules". Each module adds new capability or extends features defined in CSS 2, over preserving backward compatibility. Work on CSS level 3 started around the time of publication of the original CSS 2 recommendation. The earliest CSS 3 drafts were published in June 1999.

Due to the modularization, different modules have different stability and are in different status. As of March 2011, there are over 40 CSS modules published from the CSS Working Group. Some modules such as Selectors, Namespaces, Color and Media Queries are considered stable and are either in Candidate Recommendation or Proposed Recommendation status. On 7 June 2011, the CSS 3 Color Module was published as a W3C Recommendation.

CSS Limitations

Some noted limitations of the current capabilities of CSS include:

- Poor controls for flexible layouts - While new additions to CSS 3 provide a stronger, more robust feature-set for layout, CSS is still at heart a styling language (for fonts, colours, borders and other decoration), not a layout language (for blocks with positions, sizes, margins, and so on). These limitations mean that creating fluid layouts generally requires hand-coding of CSS, and has held back the development of a standards-based WYSIWYG editor.
- Selectors are unable to ascend - CSS offers no way to select a parent or ancestor of an element that satisfies certain criteria. A more advanced selector scheme (such as XPath) would enable more sophisticated style sheets. However, the major reasons for the CSS Working Group rejecting proposals for parent selectors are related to browser performance and incremental rendering issues.
- Vertical control limitations - While horizontal placement of elements is generally easy to control, vertical placement is frequently unintuitive, convoluted, or impossible. Simple tasks, such as cantering an element vertically or getting a footer to be placed no higher than bottom of viewport, either require complicated and unintuitive style rules, or simple but widely unsupported rules.
- Absence of expressions - There is currently no ability to specify property values as simple expressions (such as margin-left: 10% – 3em + 4px;). This would be useful in a variety of cases, such as calculating the size of columns subject to a constraint on the sum of all columns. However, a working draft with a calc() value to address this limitation has been published by the CSS WG.
- Lack of column declaration - While possible in current CSS 3 (using the column-count module), layouts with multiple columns can be complex to implement in CSS 2.1. With CSS 2.1, the process is often done using floating elements, which are often rendered differently by

different browsers, different computer screen shapes, and different screen ratios set on standard monitors.

- Cannot explicitly declare new scope independently of position - Scoping rules for properties such as z-index look for the closest parent element with a position:absolute or position:relative attribute. This odd coupling has undesired effects such as it is impossible to avoid declaring a new scope when one is forced to adjust an element's position, preventing one from using the desired scope of a parent element.
- Pseudo-class dynamic behavior not controllable - CSS implements pseudo-classes that allow a degree of user feedback by conditional application of alternate styles. One CSS pseudo-class, ":hover", is dynamic (equivalent of javascript "onmouseover") and has potential for abuse (e.g., implementing cursor-proximity popups), but CSS has no ability for a client to disable it (no "disable"-like property) or limit its effects (no "nochange"-like values for each property).

Advantages of CSS

- Flexibility - By combining CSS with the functionality of a Content Management System, a considerable amount of flexibility can be programmed into content submission forms. This allows a contributor, who may not be familiar or able to understand or edit CSS or HTML code to select the layout of an article or other page they are submitting on-the-fly, in the same form. When working with large-scale, complex sites, with many contributors such as news and informational sites, this advantage weighs heavily on the feasibility and maintenance of the project.
- Accessibility - Without CSS, web designers must typically lay out their pages with techniques that hinder accessibility for vision-impaired users, like HTML tables
- Separation of content from presentation - CSS facilitates publication of content in multiple presentation formats based on nominal parameters. Nominal parameters include explicit user preferences, different web browsers, the type of device being used to view the content (a desktop computer or mobile Internet device), the geographic location of the user and many other variables.
- Site-wide consistency - When CSS is used effectively, in terms of inheritance and "cascading," a global style sheet can be used to affect and style elements site-wide. If the situation arises that the styling of the elements should need to be changed or adjusted, these changes can be made by editing rules in the global style sheet. Before CSS, this sort of maintenance was more difficult, expensive and time-consuming.
- Bandwidth - A stylesheet, whether internal to the source document or separate, will specify the style once for a range of HTML elements selected by class, type or relationship to others.

This is much more efficient than repeating style information inline for each occurrence of the element. An external stylesheet is usually stored in the browser cache, and can therefore be used on multiple pages without being reloaded, further reducing data transfer over a network.

- Page reformatting - With a simple change of one line, a different style sheet can be used for the same page. This has advantages for accessibility, as well as providing the ability to tailor a page or site to different target devices. Furthermore, devices not able to understand the styling still display the content.

Styles Solved a Big Problem

HTML was never intended to contain tags for formatting a document. HTML was intended to define the content of a document, like:

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

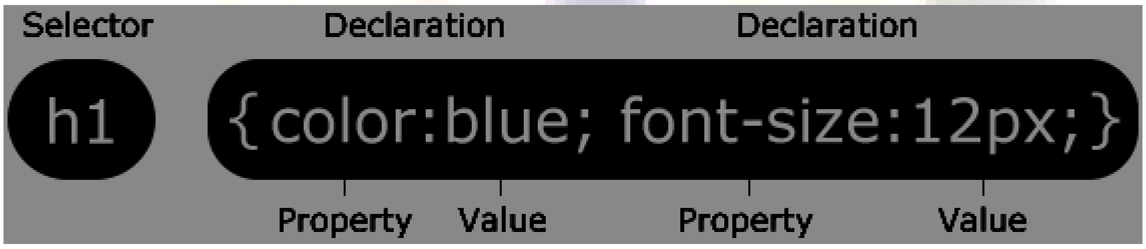
When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large web sites, where fonts and color information were added to every single page, became a long and expensive process. To solve this problem, the World Wide Web Consortium (W3C) created CSS. In HTML 4.0, all formatting could be removed from the HTML document, and stored in a separate CSS file. All browsers support CSS today.

CSS Saves a Lot of Work

CSS defines HOW HTML elements are to be displayed. Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in a Web site, just by editing one single file!

CSS Syntax

A CSS rule has two main parts: a selector, and one or more declarations:



The *selector* is normally the HTML element you want to style.

Each *declaration* consists of a property and a value.

The *property* is the style attribute you want to change. Each property has a *value*.

CSS Example

A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly brackets:

p {color:red;text-align:center;}

To make the CSS more readable, you can put one declaration on each line, like this:

```
p
{
color:red;
text-align:center;
}
```

CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers. A CSS comment begins with "/*", and ends with "*/", like this:

```
/*This is a comment*/
p
{
text-align:center;
/*This is another comment*/
color:black;
font-family:arial;
}
```

CSS Id and Class

In addition to setting a style for a HTML element, CSS allows you to specify your own selectors called "id" and "class".

The id Selector

The id selector is used to specify a style for a single, unique element. The id selector uses the id attribute of the HTML element, and is defined with a "#". The style rule below will be applied to the element with id="para1":

Example

```
#para1

{

text-align:center;

color:red;

}
```

Note: Do NOT start an ID name with a number! It will not work in Mozilla/Firefox.

The class Selector

The class selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements. This allows you to set a particular style for many HTML elements with the same class. The class selector uses the HTML class attribute, and is defined with a "." In the example below, all HTML elements with class="center" will be center-aligned:

Example

```
.center {text-align:center;}
```

You can also specify that only specific HTML elements should be affected by a class. In the example below, all p elements with class="center" will be center-aligned:

Example

```
p.center {text-align:center;}
```

Note: Do NOT start a class name with a number! This is only supported in Internet Explorer.

Three Ways to Insert CSS

When a browser reads a style sheet, it will format the document according to it. There are three ways of inserting a style sheet:

1. External style sheet
2. Internal style sheet
3. Inline style

1. *External Style Sheet*

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

```
<head>

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

</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a .css extension. An example of a style sheet file is shown below:

```
hr {color:sienna;}

p {margin-left:20px;}

body {background-image:url("images/back40.gif");}
```

2. *Internal Style Sheet*

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, by using the <style> tag, like this:

```
<head>

<style type="text/css">

hr {color:sienna;}

p {margin-left:20px;}

body {background-image:url("images/back40.gif");}
```

```
</style>

</head>
```

3. Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly!

To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet. For example, an external style sheet has these properties for the h3 selector:

```
h3

{

color:red;

text-align:left;

font-size:8pt;

}
```

And an internal style sheet has these properties for the h3 selector:

```
h3

{

text-align:right;

font-size:20pt;

}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

color:red;

text-align:right;

font-size:20pt;

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

Multiple Styles Will Cascade into One

Styles can be specified:

- inside an HTML element
- inside the head section of an HTML page
- in an external CSS file

Note: Even multiple external style sheets can be referenced inside a single HTML document.

Cascading order

What style will be used when there is more than one style specified for an HTML element? Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

1. Browser default
2. External style sheet
3. Internal style sheet (in the head section)
4. Inline style (inside an HTML element)

So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or in a browser (a default value).

Note: If the link to the external style sheet is placed after the internal style sheet in HTML <head>, the external style sheet will override the internal style sheet!

5.2 CSS Background

CSS background properties are used to define the background effects of an element. CSS properties used for background effects:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

Background Color

The background-color property specifies the background color of an element. The background color of a page is defined in the body selector:

Example

body {background-color:#b0c4de;}

With CSS, a color is most often specified by:

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

In the example below, the h1, p, and div elements have different background colors:

Example

h1 {background-color:#6495ed;}

p {background-color:#e0ffff;}

div {background-color:#b0c4de;}

Background Image

The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element. The background image for a page can be set like this:

Example

```
body {background-image:url('paper.gif');}
```

Background Image - Repeat Horizontally or Vertically

By default, the background-image property repeats an image both horizontally and vertically. If the image is repeated only horizontally (repeat-x):

Example

```
body

{

background-image:url('gradient2.png');

background-repeat:repeat-x;

}
```

Background Image - Set position and no-repeat

Note: When using a background image, use an image that does not disturb the text. Showing the image only once is specified by the background-repeat property:

Example

```
body

{

background-image:url('img_tree.png');

background-repeat:no-repeat;

}
```

The position of the image is specified by the background-position property:

Example

```
body

{

background-image:url('img_tree.png');

background-repeat:no-repeat;

background-position:right top;

}
```

Background - Shorthand property

As you can see from the examples above, there are many properties to consider when dealing with backgrounds. To shorten the code, it is also possible to specify all the properties in one single property. This is called a shorthand property. The shorthand property for background is simply "background":

Example

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

When using the shorthand property the order of the property values are:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

It does not matter if one of the property values is missing, as long as the ones that are present are in this order.

5.3 CSS Text

Text Color

The color property is used to set the color of the text. The default color for a page is defined in the body selector.

Example

body {color:blue;}

h1 {color:#00ff00;}

h2 {color:rgb(255,0,0);}

Note: For W3C compliant CSS: If you define the color property, you must also define the background-color property.

Text Alignment

The text-align property is used to set the horizontal alignment of a text. Text can be centered, or aligned to the left or right, or justified. When text-align is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

Example

h1 {text-align:center;}

p.date {text-align:right;}

p.main {text-align:justify;}

Text Decoration

The text-decoration property is used to set or remove decorations from text. The text-decoration property is mostly used to remove underlines from links for design purposes:

Example

a {text-decoration:none;}

It can also be used to decorate text:

Example

h1 {text-decoration:overline;}

h2 {text-decoration:line-through;}

h3 {text-decoration:underline;}

h4 {text-decoration:blink;}

Note: It is not recommended to underline text that is not a link, as this often confuses users.

Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

Example

p.uppercase {text-transform:uppercase;}

p.lowercase {text-transform:lowercase;}

p.capitalize {text-transform:capitalize;}

Text Indentation

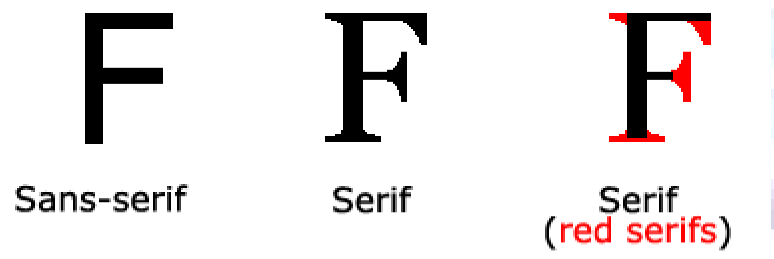
The text-indentation property is used to specify the indentation of the first line of a text.

Example

p {text-indent:50px;}

5.4 CSS Font

CSS font properties define the font family, boldness, size, and the style of a text.



On computer screens, sans-serif fonts are considered easier to read than serif fonts.

CSS Font Families

In CSS, there are two types of font family names:

- generic family - a group of font families with a similar look (like "Serif" or "Monospace")
- font family - a specific font family (like "Times New Roman" or "Arial")

Generic family	**Font family**	**Description**
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

Font Family

The font family of a text is set with the font-family property. The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font. Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like font-family: "Times New Roman".

More than one font family is specified in a comma-separated list:

Example

```
p{font-family:"Times New Roman", Times, serif;}
```

For more commonly used font combinations, look at our Web Safe Font Combinations.

Font Style

The font-style property is mostly used to specify italic text. This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example

```
p.normal {font-style:normal;}

p.italic {font-style:italic;}
```

p.oblique {font-style:oblique;}

Font Size

The font-size property sets the size of the text. Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs. Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs. The font-size value can be an absolute, or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

Set Font Size With Pixels

Setting the text size with pixels, gives you full control over the text size:

Example

```
h1 {font-size:40px;}

h2 {font-size:30px;}

p {font-size:14px;}
```

The example above allows Firefox, Chrome, and Safari to resize the text, but not Internet Explorer.

The text can be resized in all browsers using the zoom tool (however, this resizes the entire page, not just the text).

Set Font Size With Em

To avoid the resizing problem with Internet Explorer, many developers use em instead of pixels. The em size unit is recommended by the W3C. 1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px. The size can be calculated from pixels to em using this formula: pixels/16=em

Example

```
h1 {font-size:2.5em;} /* 40px/16=2.5em */

h2 {font-size:1.875em;} /* 30px/16=1.875em */

p {font-size:0.875em;} /* 14px/16=0.875em */
```

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

5.5 CSS Links

Links can be styled in different ways. Links can be styled with any CSS property (e.g. color, font-family, background, etc.). Special for links are that they can be styled differently depending on what state they are in. The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

Example

```
a:link {color:#FF0000;}      /* unvisited link */
a:visited {color:#00FF00;}  /* visited link */
a:hover {color:#FF00FF;}  /* mouse over link */
a:active {color:#0000FF;}  /* selected link */
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Common Link Styles

In the example above the link changes color depending on what state it is in.

Text Decoration

The text-decoration property is mostly used to remove underlines from links:

Example

a:link {text-decoration:none;}

a:visited {text-decoration:none;}

a:hover {text-decoration:underline;}

a:active {text-decoration:underline;}

Background Color

The background-color property specifies the background color for links:

Example

a:link {background-color:#B2FF99;}

a:visited {background-color:#FFFF85;}

a:hover {background-color:#FF704D;}

a:active {background-color:#FF704D;}

5.6 CSS Lists

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker

List

In HTML, there are two types of lists:

- unordered lists - the list items are marked with bullets
- ordered lists - the list items are marked with numbers or letters

With CSS, lists can be styled further, and images can be used as the list item marker.

Different List Item Markers

The type of list item marker is specified with the list-style-type property:

Example

ul.a {list-style-type: circle;}

ul.b {list-style-type: square;}

ol.c {list-style-type: upper-roman;}

ol.d {list-style-type: lower-alpha;}

An Image as The List Item Marker

To specify an image as the list item marker, use the list-style-image property:

Example

```
ul
{
list-style-image: url('sqpurple.gif');
}
```

The example above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari.
If you want the image-marker to be placed equally in all browsers, a crossbrowser solution is explained below.
Example

```
ul
{
list-style-type: none;
padding: 0px;
margin: 0px;
}
li
{
background-image: url(sqpurple.gif);
background-repeat: no-repeat;
background-position: 0px 5px;
padding-left: 14px;
}
```

Example explained:

- For ul:
  - Set the list-style-type to none to remove the list item marker
  - Set both padding and margin to 0px (for cross-browser compatibility)
- For li:
  - Set the URL of the image, and show it only once (no-repeat)
  - Position the image where you want it (left 0px and down 5px)
  - Position the text in the list with padding-left

List - Shorthand property

It is also possible to specify all the list properties in one, single property. This is called a shorthand property. The shorthand property used for lists, is the list-style property:

Example

```
ul
{
list-style: square url("sqpurple.gif");
}
```

When using the shorthand property, the order of the values are:

- list-style-type
- list-style-position
- list-style-image

It does not matter if one of the values above are missing, as long as the rest are in the specified

5.7 CSS Tables

The look of an HTML table can be greatly improved with CSS.

Table Borders

To specify table borders in CSS, use the border property. The example below specifies a black border for table, th, and td elements:

Example

```
table, th, td
{
border: 1px solid black;
}
```

Notice that the table in the example above has double borders. This is because both the table, th, and td elements have separate borders. To display a single border for the table, use the border-collapse property.

Collapse Borders

The border-collapse property sets whether the table borders are collapsed into a single border or separated:

Example

```
table
{
border-collapse:collapse;
}
```

```
table,th, td
{
border: 1px solid black;
}
```

Table Width and Height

Width and height of a table is defined by the width and height properties. The example below sets the width of the table to 100%, and the height of the th elements to 50px:

Example

```
table
{
width:100%;
}
th
{
height:50px;
}
```

Table Text Alignment

The text in a table is aligned with the text-align and vertical-align properties. The text-align property sets the horizontal alignment, like left, right, or center:

Example

```
td
{
text-align:right;
}
```

The vertical-align property sets the vertical alignment, like top, bottom, or middle:

Example

```
td
{
height:50px;
vertical-align:bottom;
}
```

Table Padding

To control the space between the border and content in a table, use the padding property on td and th elements:

Example

```
td
{
padding:15px;
}
```

Table Color

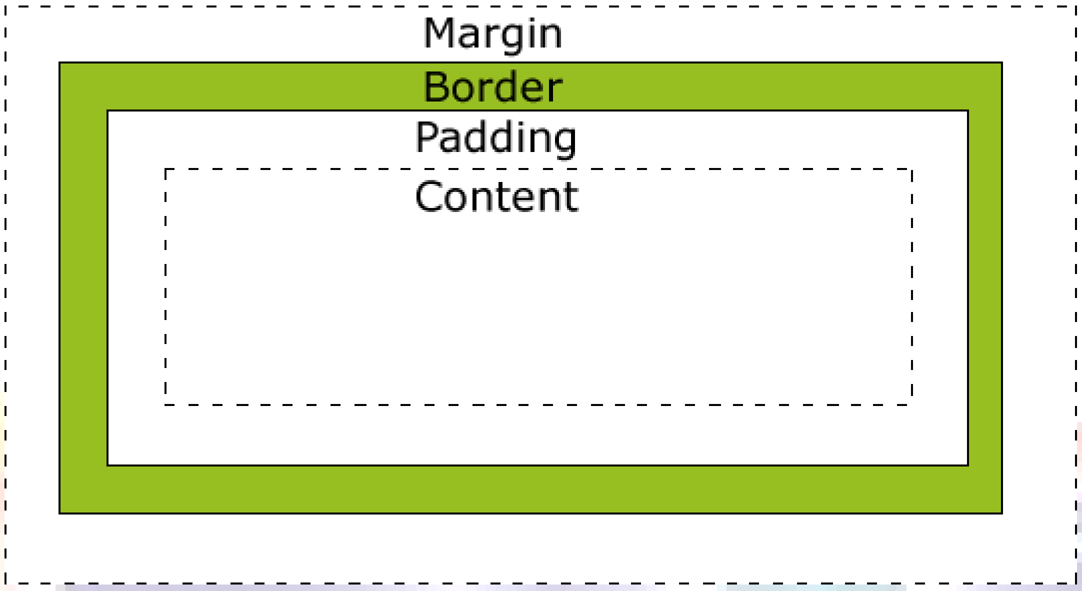
The example below specifies the color of the borders, and the text and background color of th *elements*:

Example

```
table, td, th
{
border:1px solid green;
}
th
{
background-color:green;
color:white;
}
```

5.8 The CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout. The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content. The box model allows us to place a border around elements and space elements in relation to other elements. The image below illustrates the box model:



Explanation of the different parts:

- Margin - Clears an area around the border. The margin does not have a background color, it is completely transparent
- Border - A border that goes around the padding and content. The border is affected by the background color of the box
- Padding - Clears an area around the content. The padding is affected by the background color of the box
- Content - The content of the box, where text and images appear

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Width and Height of an Element

When you specify the width and height properties of an element with CSS, you are just setting the width and height of the content area. To know the full size of the element, you must also add the padding, border and margin. The total width of the element in the example below is 300px:

```
width:250px;
padding:10px;
border:5px solid gray;
margin:10px;
```

Let's do the math:

250px (width)

+ 20px (left and right padding)

+ 10px (left and right border)

+ 20px (left and right margin)

= 300px

Imagine that you only had 250px of space. Let's make an element with a total width of 250px:

Example

```
width:220px;
padding:10px;
border:5px solid gray;
margin:0px;
```

The total width of an element should always be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should always be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Browsers Compatibility Issue

If you tested the previous example in Internet Explorer, you saw that the total width was not exactly 250px. IE8 and earlier versions includes padding and border in the width, when the width property is set, unless a DOCTYPE is declared. To fix this problem, just add a DOCTYPE to the code:

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
div.ex
{
width:220px;
padding:10px;
border:5px solid gray;
margin:0px;
}
```

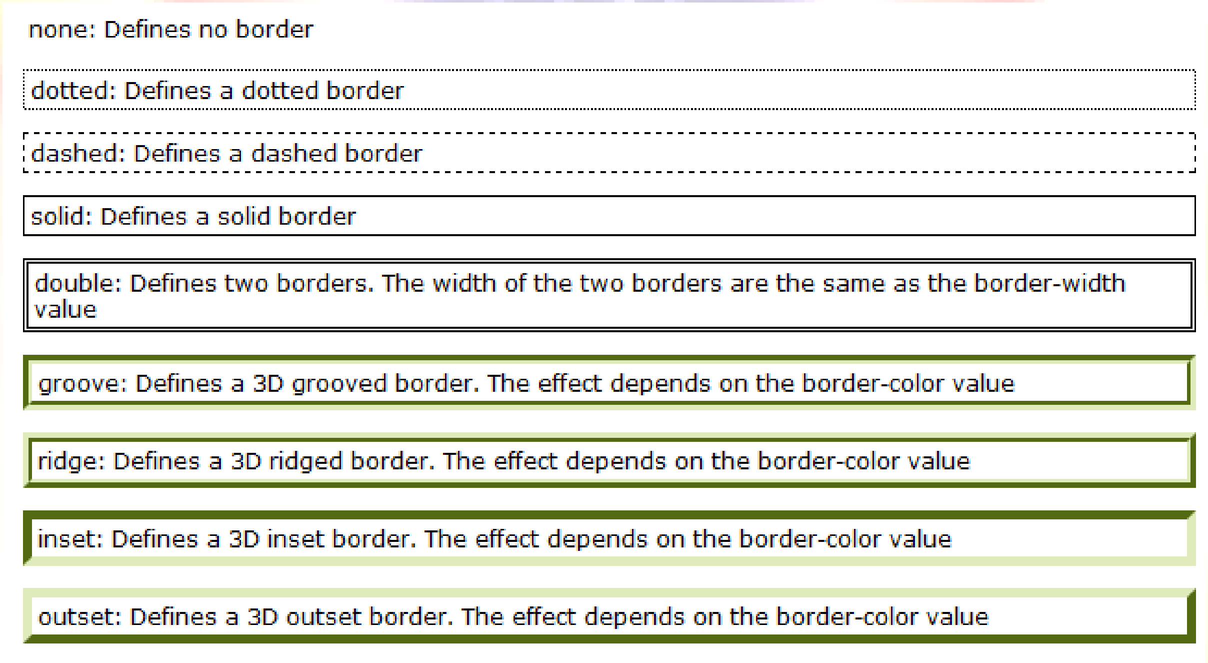
```
</style>
</head>
```

CSS Border Properties

The CSS border properties allow you to specify the style and color of an element's border.

Border Style

The border-style property specifies what kind of border to display. None of the border properties will have ANY effect unless the border-style property is set. border-style values:



none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

groove: Defines a 3D grooved border. The effect depends on the border-color value

ridge: Defines a 3D ridged border. The effect depends on the border-color value

inset: Defines a 3D inset border. The effect depends on the border-color value

outset: Defines a 3D outset border. The effect depends on the border-color value

Border Width

The border-width property is used to set the width of the border. The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick. Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

```
p.one
{
border-style:solid;
border-width:5px;
}
p.two
{
border-style:solid;
border-width:medium;
}
```

Border Color

The border-color property is used to set the color of the border. Note: The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

```
p.one
{
border-style:solid;
border-color:red;
}
p.two
{
border-style:solid;
border-color:#98bf21;
}
```

Border - Individual sides

In CSS it is possible to specify different borders for different sides:

Example

```
p
{
border-top-style:dotted;
border-right-style:solid;
border-bottom-style:dotted;
border-left-style:solid;
}
```

The example above can also be set with a single property:

Example

```
border-style:dotted solid;
```

The border-style property can have from one to four values.

- border-style:dotted solid double dashed;
    - top border is dotted
    - right border is solid
    - bottom border is double

    - left border is dashed
- border-style:dotted solid double;
    - top border is dotted
    - right and left borders are solid
    - bottom border is double
- border-style:dotted solid;
    - top and bottom borders are dotted
    - right and left borders are solid
- border-style:dotted;
    - all four borders are dotted

The border-style property is used in the example above. However, it also works with border-width and border-color.

Border - Shorthand property

To shorten the code, it is also possible to specify all the border properties in one property. This is called a shorthand property. The shorthand property for the border properties is "border":

Example

border:5px solid red;

When using the border property, the order of the values are:

- border-width
- border-style
- border-color

It does not matter if one of the values above are missing (although, border-style is required), as long as the rest are in the specified order.

CSS Outlines

An outline is a line that is drawn around elements, outside the border edge, to make the element "stand out". The outline properties specifies the style, color, and width of an outline. However, it is different from the border property. The outline is not a part of the element's dimensions, therefore the element's width and height properties do not contain the width of the outline.

All CSS Outline Properties

The number in the "CSS" column indicates in which CSS version the property is defined (CSS1 or CSS2).

Property	**Description**	**Values**	**CSS**
outline	Sets all the outline properties in one	*outline-color*	2

	declaration	*outline-style* *outline-width* inherit	
outline-color	Sets the color of an outline	*color_name* *hex_number* *rgb_number* invert inherit	2
outline-style	Sets the style of an outline	none dotted dashed solid double groove ridge inset outset inherit	2
outline-width	Sets the width of an outline	thin medium thick *length* inherit	2

CSS Margin

The CSS margin properties define the space around elements. The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent. The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Possible Values

Value	**Description**
auto	The browser sets the margin. The result of this is dependant of the browser
length	Defines a fixed margin (in pixels, pt, em, etc.)
%	Defines a margin in % of the containing element

It is possible to use negative values, to overlap content.

Margin - Individual sides

In CSS, it is possible to specify different margins for different sides:

Example

margin-top:100px;

margin-bottom:100px;

margin-right:50px;

margin-left:50px;

Margin - Shorthand property

To shorten the code, it is possible to specify all the margin properties in one property. This is called a shorthand property. The shorthand property for all the margin properties is "margin":

Example

margin:100px 50px;

The margin property can have from one to four values.

- margin:25px 50px 75px 100px;
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px
- margin:25px 50px 75px;
  - top margin is 25px
  - right and left margins are 50px
  - bottom margin is 75px
- margin:25px 50px;
  - top and bottom margins are 25px
  - right and left margins are 50px
- margin:25px;
  - all four margins are 25px

CSS Padding

The CSS padding properties define the space between the element border and the element content. The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element. The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all paddings at once.

Possible Values

Value	**Description**
length	Defines a fixed padding (in pixels, pt, em, etc.)
%	Defines a padding in % of the containing element

Padding - Individual sides

In CSS, it is possible to specify different padding for different sides:

Example

padding-top:25px;

padding-bottom:25px;

padding-right:50px;

padding-left:50px;

Padding - Shorthand property

To shorten the code, it is possible to specify all the padding properties in one property. This is called a shorthand property. The shorthand property for all the padding properties is "padding":

Example

padding:25px 50px;

The padding property can have from one to four values.

- padding:25px 50px 75px 100px;
    - top padding is 25px
    - right padding is 50px
    - bottom padding is 75px
    - left padding is 100px

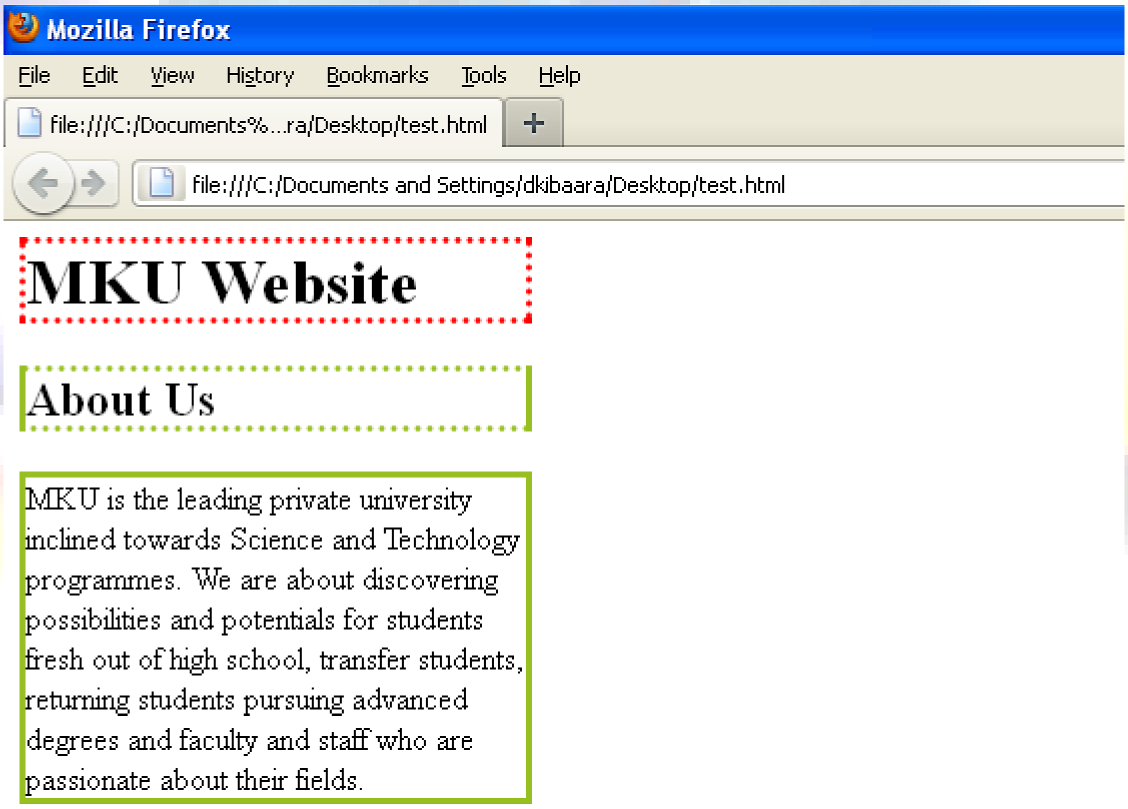
- padding:25px 50px 75px;
    - top padding is 25px
    - right and left paddings are 50px
    - bottom padding is 75px

- padding:25px 50px;
    - top and bottom paddings are 25px
    - right and left paddings are 50px

- padding:25px;
    - all four paddings are 25px

Chapter Review Questions

1. Define what is CSS
2. Outline the advantages of CSS
3. Giving examples can you explain three ways to insert CSS into a HTML document
4. Write CSS code to add a background image to a HTML document
5. Write CSS code to display the following box model


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Documents%...ra/Desktop/test.html
file:///C:/Documents and Settings/dkibaara/Desktop/test.html

MKU Website

About Us

MKU is the leading private university inclined towards Science and Technology programmes. We are about discovering possibilities and potentials for students fresh out of high school, transfer students, returning students pursuing advanced degrees and faculty and staff who are passionate about their fields.

Suggested Further Reading

1. Niederst J., Learning Web design, Shroff Publishers
2. Peter, K., Web design, Tools and Techniques, Peach Pit press
3. http://www.w3schools.com

CHAPTER SIX: INTRODUCTION TO JAVASCRIPT

Learning Objectives:

By the end of this chapter the learner shall be able to;

i. explain what is JavaScript

ii. create JavaScript documents

iii. explain what are conditional statements and loops

6.1 Introduction to JavaScript

JavaScript is a prototype-based, object-oriented scripting language that is dynamic, weakly typed and has first-class functions. It is also considered a functional programming language like Scheme and OCaml because it has closures and supports higher-order functions. JavaScript is an implementation of the ECMAScript language standard and is primarily used in the form of client-side JavaScript, implemented as part of a web browser in order to provide enhanced user interfaces and dynamic websites. This enables programmatic access to computational objects within a host environment.

JavaScript uses syntax influenced by that of C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages. JavaScript is used in billions of Web pages to add functionality, validate forms, communicate with the server, and much more. JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.

JavaScript:

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages

- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Difference between Java and JavaScript

Java and JavaScript are two completely different languages in both concept and design. Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

JavaScript uses

- JavaScript gives HTML designers a programming tool - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- JavaScript can react to events - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element
- JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- JavaScript can be used to detect the visitor's browser - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- JavaScript can be used to create cookies - A JavaScript can be used to store and retrieve information on the visitor's computer

JavaScript = ECMAScript

JavaScript is an implementation of the ECMAScript language standard. ECMA-262 is the official JavaScript standard. JavaScript was invented by Brendan Eich at Netscape (with Navigator 2.0), and has appeared in all browsers since 1996.The official standardization was adopted by the ECMA organization (an industry standardization association) in 1997. The ECMA standard (called ECMAScript-262) was approved as an international ISO (ISO/IEC 16262) standard in 1998. The development is still in progress.

Writing to The HTML Document

The HTML <script> tag is used to insert a JavaScript into an HTML page.

Example

The example below writes a <p> element with current date information to the HTML document:

```
<html>

<body>

<h1>My First Web Page</h1>

<script type="text/javascript">

document.write("<p>" + Date() + "</p>");

</script>

</body>

</html>
```

Changing HTML Elements

The example below writes the current date into an existing <p> element:

Example

```
<html>

<body>

<h1>My First Web Page</h1>

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

<script type="text/javascript">

document.getElementById("demo").innerHTML=Date();

</script>

</body>

</html>
```

Note: To manipulate HTML elements JavaScript uses the DOM (Document Object Model) method getElementById(). This method accesses the element with the specified id.

To insert a JavaScript into an HTML page, use the <script> tag. Inside the <script> tag use the type attribute to define the scripting language. The <script> and </script> tells where the JavaScript starts and ends:

```
<html>

<body>

<h1>My First Web Page</h1>

<p id="demo">This is a paragraph.</p>

<script type="text/javascript">

... some JavaScript code ...

</script>

</body>

</html>
```

The lines between the <script> and </script> contain the JavaScript and are executed by the browser. In this case the browser will replace the content of the HTML element with id="demo", with the current date:

```
<html>

<body>

<h1>My First Web Page</h1>

<p id="demo">This is a paragraph.</p>

<script type="text/javascript">

document.getElementById("demo").innerHTML=Date();

</script>

</body>
```

</html>

Without the <script> tag(s), the browser will treat "document.getElementById("demo").innerHTML=Date();" as pure text and just write it to the page

6.2 JavaScript Comments

JavaScript comments can be used to make the code more readable. Comments can be added to explain the JavaScript, or to make the code more readable.

Single line comments start with //.

The following example uses single line comments to explain the code:

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

// Write a heading

document.write("<h1>This is a heading</h1>");

// Write two paragraphs:

document.write("<p>This is a paragraph.</p>");

document.write("<p>This is another paragraph.</p>");

</script>
```

JavaScript Multi-Line Comments

Multi line comments start with /* and end with */. The following example uses a multi line comment to explain the code:

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

/*

The code below will write

one heading and two paragraphs

*/

document.write("<h1>This is a heading</h1>");

document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");

</script>
```

Using Comments to Prevent Execution

In the following example the comment is used to prevent the execution of a single code line (can be suitable for debugging):

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

//document.write("<h1>This is a heading</h1>");

document.write("<p>This is a paragraph.</p>");

document.write("<p>This is another paragraph.</p>");

</script>
```

In the following example the comment is used to prevent the execution of a code block (can be suitable for debugging):

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

/*

document.write("<h1>This is a heading</h1>");

document.write("<p>This is a paragraph.</p>");

document.write("<p>This is another paragraph.</p>");

*/

</script>
```

Using Comments at the End of a Line

In the following example the comment is placed at the end of a code line:

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

document.write("Hello"); // Write "Hello"

document.write(" Dolly!"); // Write " Dolly!"
```

```
</script>
```

Some Browsers do Not Support JavaScript

Browsers that do not support JavaScript, will display JavaScript as page content. To prevent them from doing this, and as a part of the JavaScript standard, the HTML comment tag should be used to "hide" the JavaScript. Just add an HTML comment tag <!-- before the first JavaScript statement, and a --> (end of comment) after the last JavaScript statement, like this:

```
<html>

<body>

<script type="text/javascript">

<!--

document.getElementById("demo").innerHTML=Date();

//-->

</script>

</body>

</html>
```

The two forward slashes at the end of comment line (//) is the JavaScript comment symbol. This prevents JavaScript from executing the --> tag.

6.3 Where to place JavaScript in HTML document

- In the head section
- In the body section
- Using External JavaScript

1. ***JavaScript in <body>***

The example below writes the current date into an existing <p> element when the page loads:

```
<html>

<body>
```

```
<h1>My First Web Page</h1>

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

<script type="text/javascript">

document.getElementById("demo").innerHTML=Date();

</script>

</body>

</html>
```

Note that the JavaScript is placed at the bottom of the page to make sure it is not executed before the <p> element is created.

2. *JavaScript in <head>*

The example below calls a function when a button is clicked:

```
<html>

<head>

<script type="text/javascript">

function displayDate()

{

document.getElementById("demo").innerHTML=Date();

}

</script>

</head>

<body>

<h1>My First Web Page</h1>

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

<button type="button" onclick="displayDate()">Display Date</button>
```

```
</body>

</html>
```

3. ***Scripts in <head> and <body>***

You can place an unlimited number of scripts in your document, and you can have scripts in both the body and the head section at the same time. It is a common practice to put all functions in the head section, or at the bottom of the page. This way they are all in one place and do not interfere with page content.

4. ***Using an External JavaScript***

JavaScript can also be placed in external files. External JavaScript files often contain code to be used on several different web pages. External JavaScript files have the file extension .js.

Note: External script cannot contain the <script></script> tags!

To use an external script, point to the .js file in the "src" attribute of the <script> tag:

```
<html>

<head>

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

</head>

<body>

</body>

</html>
```

6.4 JavaScript Statements

JavaScript is a sequence of statements to be executed by the browser. A JavaScript statement is a command to a browser. The purpose of the command is to tell the browser what to do. This JavaScript statement tells the browser to write "Hello Dolly" to the web page:

```
document.write("Hello Dolly");
```

It is normal to add a semicolon at the end of each executable statement. The semicolon is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. Note: Using semicolons makes it possible to write multiple statements on one line.

JavaScript is Case Sensitive

Unlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.

JavaScript Code

JavaScript code (or just JavaScript) is a sequence of JavaScript statements. Each statement is executed by the browser in the sequence they are written. This example will write a heading and two paragraphs to a web page:

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

document.write("<h1>This is a heading</h1>");

document.write("<p>This is a paragraph.</p>");

document.write("<p>This is another paragraph.</p>");

</script>
```

JavaScript Blocks

JavaScript statements can be grouped together in blocks. Blocks start with a left curly bracket {, and end with a right curly bracket}. The purpose of a block is to make the sequence of statements execute together. This example will write a heading and two paragraphs to a web page:

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

{

document.write("<h1>This is a heading</h1>");

document.write("<p>This is a paragraph.</p>");

document.write("<p>This is another paragraph.</p>");

}

</script>
```

The example above is not very useful. It just demonstrates the use of a block. Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).

6.5 JavaScript Variables

Variables are "containers" for storing information. JavaScript variables are used to hold values or expressions. A variable can have a short name, like x, or a more descriptive name, like carname.

Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter or the underscore character

A variable's value can change during the execution of a script. You can refer to a variable by its name to display or change its value.

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables. You declare JavaScript variables with the var keyword:

```
var x;

var carname;
```

After the declaration shown above, the variables are empty (they have no values yet). However, you can also assign values to the variables when you declare them:

```
var x=5;

var carname="Volvo";
```

After the execution of the statements above, the variable x will hold the value 5, and carname will hold the value Volvo.

Note: When you assign a text value to a variable, use quotes around the value. If you redeclare a JavaScript variable, it will not lose its value.

Local JavaScript Variables

A variable declared within a JavaScript function becomes LOCAL and can only be accessed within that function. (the variable has local scope). You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared. Local variables are destroyed when you exit the function.

Global JavaScript Variables

Variables declared outside a function become GLOBAL, and all scripts and functions on the web page can access it. Global variables are destroyed when you close the page. If you declare a variable, without using "var", the variable always becomes GLOBAL.

Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that have not yet been declared, the variables will automatically be declared as global variables. The statements below will declare the variables x and carname as global variables (if they don't already exist):

```
x=5;

carname="Volvo";
```

6.6 JavaScript Operators

= is used to assign values.

+ is used to add values.

The assignment operator = is used to assign values to JavaScript variables.

The arithmetic operator + is used to add values together.

```
y=5;

z=2;

x=y+z;
```

The value of x, after the execution of the statements above, is 7.

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values. Given that y=5, the table below explains the arithmetic operators:

Operator	Description	Example	Result	
+	Addition	x=y+2	x=7	y=5

-	Subtraction	x=y-2	x=3	y=5
*	Multiplication	x=y*2	x=10	y=5
/	Division	x=y/2	x=2.5	y=5
%	Modulus (division remainder)	x=y%2	x=1	y=5
++	Increment	x=++y	x=6	y=6
		x=y++	x=5	y=6
--	Decrement	x=--y	x=4	y=4
		x=y--	x=5	y=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables. Given that x=10 and y=5, the table below explains the assignment operators:

Operator	**Example**	**Same As**	**Result**
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together. To add two or more string variables together, use the + operator.

txt1="What a very";

txt2="nice day";

txt3=txt1+txt2;

After the execution of the statements above, the variable txt3 contains "What a verynice day". To add a space between the two strings, insert a space into one of the strings:

txt1="What a very ";

txt2="nice day";

txt3=txt1+txt2;

or insert a space into the expression:

txt1="What a very";

txt2="nice day";

txt3=txt1+" "+txt2;

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

Adding Strings and Numbers

The rule is: If you add a number and a string, the result will be a string!

Example

```
x=5+5;
document.write(x);//output: 10

x="5"+"5";
document.write(x); output: 55

x=5+"5";
document.write(x); output: 55

x="5"+5;
document.write(x); output: 55
```

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for true or false.

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values. Given that x=5, the table below explains the comparison operators:

Operator	**Description**	**Example**
==	is equal to	x==8 is false x==5 is true
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

How comparison operators are used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

if (age<18) document.write("Too young");

Logical Operators

Logical operators are used to determine the logic between variables or values. Given that x=6 and y=3, the table below explains the logical operators:

Operator	**Description**	**Example**
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

variablename=(condition)?value1:value2

Example

greeting=(visitor=="PRES")?"Dear President ":"Dear ";

If the variable visitor has the value of "PRES", then the variable greeting will be assigned the value "Dear President " else it will be assigned "Dear".

6.7 Conditional Statements

Conditional statements are used to perform different actions based on different conditions. Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this. In JavaScript we have the following conditional statements:

- if statement - use this statement to execute some code only if a specified condition is true
- if...else statement - use this statement to execute some code if the condition is true and another code if the condition is false
- if...else if....else statement - use this statement to select one of many blocks of code to be executed
- switch statement - use this statement to select one of many blocks of code to be executed

1. ***If Statement***

Use the if statement to execute some code only if a specified condition is true.

Syntax

if (condition)

```
  {
  code to be executed if condition is true
  }
```

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

Example

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10
var d=new Date();
var time=d.getHours();
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
</script>
```

Notice that there is no ..else.. in this syntax. You tell the browser to execute some code only if the specified condition is true.

2. ***If...else Statement***

Use the, if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
  {
  code to be executed if condition is true
  }
else
  {
  code to be executed if condition is not true
  }
```

Example

```
<script type="text/javascript">
//If the time is less than 10, you will get a "Good morning" greeting.
//Otherwise you will get a "Good day" greeting.
var d = new Date();
var time = d.getHours();
```

```
if (time < 10)
  {
  document.write("Good morning!");
  }
else
  {
  document.write("Good day!");
  }
</script>
```

3. *If...else if...else Statement*

Use the, if....else if...else statement to select one of several blocks of code to be executed.

Syntax

```
if (condition1)
  {
  code to be executed if condition1 is true
  }
else if (condition2)
  {
  code to be executed if condition2 is true
  }
else
  {
  code to be executed if neither condition1 nor condition2 is true
  }
```

Example

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
else if (time>10 && time<16)
  {
  document.write("<b>Good day</b>");
  }
```

```
else
  {
  document.write("<b>Hello World!</b>");
  }
</script>
```

4. *JavaScript Switch Statement*

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is different from case 1 and 2
}
```

This is how it works: First we have a single expression n (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically.

Example

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
var d=new Date();
var theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("Finally Friday");
  break;
case 6:
  document.write("Super Saturday");
  break;
case 0:
  document.write("Sleepy Sunday");
  break;
default:
  document.write("I'm looking forward to this weekend!");
}
</script>
```

6.8 JavaScript Loops

Loops execute a block of code a specified number of times, or while a specified condition is true. Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kind of loops:

- for - loops through a block of code a specified number of times
- while - loops through a block of code while a specified condition is true

1. ***The for Loop***

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (variable=startvalue;variable<=endvalue;variable=variable+increment)
{
code to be executed
}
```

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

2. JavaScript While Loop

The while loop loops through a block of code while a specified condition is true.

Syntax

```
while (variable<=endvalue)
  {
  code to be executed
  }
```

Note: The <= could be any comparing operator.

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
</script>
</body>
</html>
```

3. The do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
do
  {
  code to be executed
  }
while (variable<=endvalue);
```

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
while (i<=5);
</script>
</body>
</html>
```

6.9 JavaScript Popup Boxes

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

1. ***Alert Box***

An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
alert("sometext");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("I am an alert box!");
}
</script>
```

```
</head>
<body>
<input type="button" onclick="show_alert()" value="Show alert box" />
</body>
</html>
```

2. ***Confirm Box***

A confirm box is often used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
confirm("sometext");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
}
</script>
</head>
<body>
<input type="button" onclick="show_confirm()" value="Show confirm box" />
</body>
</html>
```

3. ***Prompt Box***

A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
prompt("sometext","defaultvalue");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
  document.write("Hello " + name + "! How are you today?");
  }
}
</script>
</head>
<body>
<input type="button" onclick="show_prompt()" value="Show prompt box" />
</body>
</html>
```

6.10 JavaScript Functions

A function will be executed by an event or by a call to the function. To keep the browser from executing a script when the page loads, you can put your script into a function. A function contains code that will be executed by an event or by a call to the function. You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax

```
function functionname(var1,var2,...,varX)
{
some code
}
```

The parameters i.e. var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name. Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

JavaScript Function Example

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```

If the line: alert("Hello world!!") in the example above had not been put within a function, it would have been executed as soon as the page was loaded. Now, the script is not executed before a user hits the input button. The function displaymessage() will be executed if the input button is clicked.

The return Statement

The return statement is used to specify the value that is returned from the function. So, functions that are going to return a value must use the return statement. The example below returns the product of two numbers (a and b):

Example

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
```

```
return a*b;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3));
</script>
</body>
</html>
```

The Lifetime of JavaScript Variables

If you declare a variable, using "var", within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared. If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

6.11 JavaScript Break and Continue Statements

The break Statement

The break statement will break the loop and continue executing the code that follows after the loop (if any).

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    break;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
```

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

The continue Statement

The continue statement will break the current loop and continue with the next value.

Example

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    continue;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
</script>
</body>
</html>
```

JavaScript For...In Statement

The for...in statement loops through the properties of an object.

Syntax

```
for (variable in object)
  {
  code to be executed
  }
```

Note: The code in the body of the, for...in loop is executed once for each property.

Example

Looping through the properties of an object:

```
var person={fname:"John",lname:"Doe",age:25};

for (x in person)
```

```
{
document.write(person[x] + " ");
}
```

6.12 JavaScript Events

Events are actions that can be detected by JavaScript. By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript. Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

Acting to an Event

The example below displays the date when a button is clicked:

Example

```
<html>
<head>
<script type="text/javascript">
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
</head>
<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<button type="button" onclick="displayDate()">Display Date</button>
</body>
```

</html>

onLoad and onUnload

The onLoad and onUnload events are triggered when the user enters or leaves the page. The onLoad event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information. Both the onLoad and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

onFocus, onBlur and onChange

The onFocus, onBlur and onChange events are often used in combination with validation of form fields. Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

<input type="text" size="30" id="email" onchange="checkEmail()">

onSubmit

The onSubmit event is used to validate ALL form fields before submitting it. Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

<form method="post" action="xxx.htm" onsubmit="return checkForm()">

onMouseOver

The onmouseover event can be used to trigger a function when the user mouses over an HTML element.

6.13 JavaScript - Catching Errors

When browsing Web pages on the internet, we all have seen a JavaScript alert box telling us there is a runtime error and asking "Do you wish to debug?". Error message like this may be useful for developers but not for users. When users see errors, they often leave the Web page.

JavaScript Try...Catch Statement

The try...catch statement allows you to test a block of code for errors. The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

Syntax

try

```
  {
  //Run some code here
  }
catch(err)
  {
  //Handle errors here
  }
```

Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

The example below is supposed to alert "Welcome guest!" when the button is clicked. However, there's a typo in the message() function. alert() is misspelled as adddlert(). A JavaScript error occurs. The catch block catches the error and executes a custom code to handle it. The code displays a custom error message informing the user what happened:

Example

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Error description: " + err.description + "\n\n";
  txt+="Click OK to continue.\n\n";
  alert(txt);
  }
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
```

</body>

</html>

The next example uses a confirm box to display a custom message telling users they can click OK to continue viewing the page or click Cancel to go to the homepage. If the confirm method returns false, the user clicked Cancel, and the code redirects the user. If the confirm method returns true, the code does nothing:

Example

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Click OK to continue viewing this page,\n";
  txt+="or Cancel to return to the home page.\n\n";
  if(!confirm(txt))
    {
    document.location.href="http://www.w3schools.com/";
    }
  }
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

6.14 JavaScript Throw Statement

The throw statement allows you to create an exception. The throw statement allows you to create an exception. If you use this statement together with the try...catch statement, you can control program flow and generate accurate error messages.

Syntax

throw exception

The exception can be a string, integer, Boolean or an object.

Note that throw is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

The example below determines the value of a variable called x. If the value of x is higher than 10, lower than 0, or not a number, we are going to throw an error. The error is then caught by the catch argument and the proper error message is displayed:

Example

```
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 0 and 10:","");
try
  {
  if(x>10)
    {
    throw "Err1";
    }
  else if(x<0)
    {
    throw "Err2";
    }
  else if(isNaN(x))
    {
    throw "Err3";
    }
  }
catch(er)
  {
  if(er=="Err1")
```

```
    {
    alert("Error! The value is too high");
    }
  if(er=="Err2")
    {
    alert("Error! The value is too low");
    }
  if(er=="Err3")
    {
    alert("Error! The value is not a number");
    }
  }
</script>
</body>
</html>
```

6.14 JavaScript Special Characters

In JavaScript you can add special characters to a text string by using the backslash sign.

Insert Special Characters

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string. Look at the following JavaScript code:

```
var txt="We are the so-called "Vikings" from the north.";
document.write(txt);
```

In JavaScript, a string is started and stopped with either single or double quotes. This means that the string above will be chopped to: We are the so-called

To solve this problem, you must place a backslash (\) before each double quote in "Viking". This turns each double quote into a string literal:

```
var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);
```

JavaScript will now output the proper text string: We are the so-called "Vikings" from the north.

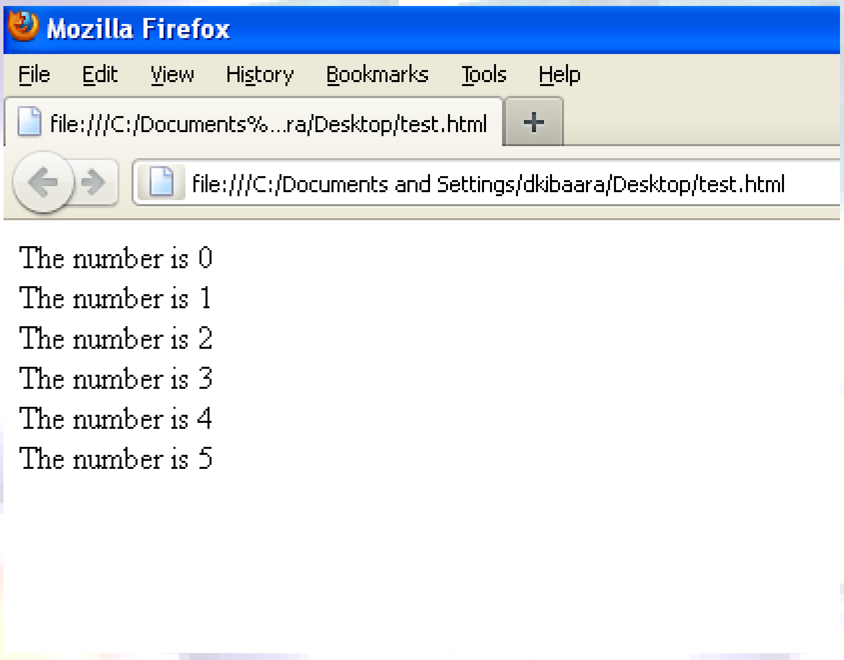
The table below lists other special characters that can be added to a text string with the backslash sign:

Code	**Outputs**
\'	single quote
\"	double quote
\\	backslash

\n	new line
\r	carriage return
\t	tab
\b	backspace
\f	form feed

Chapter Review Questions

1. Define JavaScript
2. What are the uses of JavaScript in web development
3. Write the single line comment JavaScript code
4. differentiate between local JavaScript variables and Global JavaScript variables
5. Write the JavaScript code to display the following window


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Documents%...ra/Desktop/test.html
+
file:///C:/Documents and Settings/dkibaara/Desktop/test.html

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

Using:

i. For loop

ii. While loop

Suggested Further Reading

1. Niederst J., Learning Web design, Shroff Publishers
2. Peter, K., Web design, Tools and Techniques, Peach Pit press
3. http://www.w3schools.com

SAMPLE PAPERS