Polytechnic University of the Philippines
Quezon City Branch

INSTRUCTIONAL MATERIAL FOR

WEB DEVELOPMENT

Compiled by: Prof. Cherry M. Doromal

Course Title : **Web Development**

Course Code : **COMP 20163**

Course Credit : 3 UNITS / 5 HOURS (2 Units Lecture And 1 Unit Lab)

Pre-Requisite : **INTE 30023 – Interactive Programming and Technologies I**

Course Description : This course provides a comprehensive overview of website programming. Students learn the prevailing vocabulary, tools, and standards used in the field and understand how the numerous aspects including HTML5, XHTML, CSS, JavaScript, PHP MySQL, Multimedia, scripting languages, HTTP, client servers, FTP, Web Server application, website hosting, DNS, API's, and databases function together in today's web environment. The course provides a solid web development foundation, aiming on content and client-side (browser) components (HTML5, XHTML, CSS, JavaScript, PHP MySQL, Apache, multimedia), with an overview of the server-side technologies. Students produce an interactive and dynamic website on the topic or their choice or provided by the assigned lecturer for the final project and leave the course prepared for more advanced and focused web programming lessons..

INSTITUTIONAL INTENDED LEARNING OUTCOMES (ILO)	PROGRAM INTENDED LEARNING OUTCOMES (PILO) COLLEGE		COURSE INTENDED LEARNING OUTCOMES (CILO) SUBJECT
	BSIT	BSIT Graduate Outcomes	Course Outcomes
Creative and Critical Thinking	IT01	Apply knowledge of computing, science, and mathematics appropriate to the discipline.	• Apply a structured approach to identifying needs, interests, and functionality of a website. • Design dynamic websites that meet specified needs and interests. • Write well-structured, easily maintained, standards-compliant, accessible HTML code. • Write well-structured, easily maintained, standard-compliant CSS code to present HTML pages in different ways. • Use JavaScript to add dynamic content to pages. • Select appropriate HTML, CSS, and JavaScript code from public repositories of open-source and free scripts that enhances the experience of site visitors. • Modify existing HTML, CSS, and JavaScript code to extend and alter its functionality, and to correct errors and cases of poor practice. • Write well-structured, easily maintained JavaScript code following accepted good practice. • Write JavaScript code that works in all major browsers. • Effectively debug code, making use of good practice and debugging tools.
	IT03	Analyze complex problems, and identify and define the computing requirements appropriate to its solution.	
	IT04	Identify and analyze user needs and take them into account in the selection, creation, evaluation and administration of computer based systems.	
	IT05	Design, implement, and evaluate computer based systems, processes, components, or programs to meet desired needs and requirements under various constraints.	

Creative and Critical Thinking	IT06	Integrate IT-based solutions into the user environment effectively.	
	IT09	Assist in the creation of an effective IT project plan.	
Adeptness in the Responsible Use of Technology	IT07	Apply knowledge through the use of current techniques, skills, tools and practices necessary for the IT profession	
Community Engagement	IT02	Understand best practices and standards and their applications.	
High Level of Leadership and Organizational Skills.	IT08	Function effectively as a member or leader of a development team recognizing the different roles within a team to accomplish a common goal.	
Strong Service Orientation			
Effective Communication	IT10	Communicate effectively with the computing community and with society at large about complex computing activities through logical writing, presentations, and clear instructions.	
Sense of Nationalism and Global Responsiveness.	IT11	Analyze the local and global impact of computing information technology on individuals, organizations, and society.	
Sense of Personal and Professional Ethics	IT12	Understand professional, ethical, legal, security and social issues and responsibilities in the utilization of information technology.	
Passion to Life-Long Learning	IT13	Recognize the need for and engage in planning self-l;earning and imprroving performance as a foundation for continuing professional development.	

COURSE PLAN

Week	Topic	Learning Outcomes	Methodology	Resources	Assessment
Week 1	Class orientation Discussion of course goals, expected outcomes, course policies and grading system Assigning of Groups and Officers	• Familiarize student on Outcome-Based Education • Orient the student on the course syllabus, grading system and classroom rules	Orientation Review of the syllabus, learning activities and assessment Getting to know activity Ice breaker activity Lecture/Discussion Program Demonstration Recitation/Board work	Course Syllabus	None
Week 2	**INTRODUCTION** • Internet • Intranet • World Wide Web • Web Technologies • Principles of Effective Web Design • Things to Consider in Web Designing • Support Tools for Web Creation and programming • Importance of Website	• Understand how the Internet works and the technology behind it. • Recognized and differentiate difference between Intranet and internet. • Familiarize student on world wide web • Familiarize student different websites, feature and technologies • Understand all the things to consider in web designing and programming		The Missing Link An Introduction to Web Development and Programming	Short Quiz
Week 3-4	**HTML TAGS** • Structure • Text • Lists • Links • Images • Tables • Forms • Extra Markup	• Familiarize student with the HTML structure programming and tags and elements. • Learning about markup, Headings and paragraphs, Bold, Italic, emphasis • Numbered lists, bullet lists, and definition lists. • Creating links between pages, linking to other sites and email links • Familiarize student with images, tables, forms, and extra markup HTML tags • Learn to add images to pages, choosing the right format, and optimizing images for the web • Learn to create tables and what information suits to tables. • Learn to collect information from visitors, create different kinds of form controls. • Understand different versions of HTML, elements, comments, meta information and iframes.		HTML & CSS Design and Build Websites The Missing Link An Introduction to Web Development and Programming	Hands On Programming rt Quiz

Week	Topic	Learning Outcomes	Methodology	Resources	Assessment
Week 5-6	**HTML TAGS & CSS** • Flash, Video & Audio • Introduction to CSS • Color • Text • Boxes • Lists, Tables & Forms • Layout • Images • Process and Design	• Learn to add flash movies, video and audio to a website. • Understand what CSS does and how it works. • Learn the rules, properties and values of CSS • Learn how to specify colors, color terminology, contrast, and background color. • Learn to use size and typeface of text, bold, italics, capitals, underlines, spacing between lines, words, and letters. • Learn to control size of boxes, box model for borders, margin and padding, displaying and hiding boxes. • Learn to specify bullet point styles, adding borders and backgrounds to tables, and changing the appearance of form elements • Learn to control the position of elements, creating site layouts, and designing for different sized screens • Learn to control size of images in CSS, aligning images in CSS, and adding background images • Familiarize student on how to approach building a site, understand the audience and their needs and how to present information visitors want to see		HTML & CSS Design and Build Websites The Missing Link An Introduction to Web Development and Programming	Hands On Programming Short Quiz
Week 7-8	**INTRODUCTION TO JAVASCRIPT & DOM PROGRAMMING** • JavaScript Basics • JavaScript Functions • JavaScript Events	• Learn the structure and syntax, and script coding, variables, operators, conditional statements • Understand on how to use functions such as popup boxes, function parameters, calling function, nested function, and return statements • Learn JavaScript events to execute JS coded responses and develop interactive sites. • Understand Document Object Model (DOM)		https://www.tutorialspoint.com/javascript/	Hands On Programming Short Quiz
Week 9	**MIDTERM EXAM**				Midterm Examination

Week	Topic	Learning Outcomes	Methodology	Resources	Assessment
Week 10	**INTRODUCTION TO APACHE and PHP MYSQL** • Installation of XAMPP • Introduction to PHPMyAdmin, FTP and Apache services	• Learn the concept of XAMPP services and techniques to create database, use of FTP and web server application		https://www.apachefriends.org/index.html	Hands On Exercises Short Quiz
Week 11	**FINAL PROJECT IDENTIFYING & SCHEDULING**	• Understand the requirements of the project (contents, pages, interface, and deadlines)			
Week 12-13	**PHP MySQL LECTURE** • Introduction to PHP MySQL programming structure • Variables • Connection to DBMS • Create Database, Table • Insert • Delete • Update Form application	• Learn how to build a real-world website from implementing PHP and MySQL. • Understand the requirements in building out a website through identifying different tools (hardware for web server, operating system, web server software, dbms, and programming or scripting language) • Learn how to create database and tables through programming codes. • Learn how to insert, update, and delete records from database and apply to HTML form.		PHP and MySQL Web Development (5th Edition) http://ebook-dl.com/book/3826	Hands On Programming Short Quiz
Week 14	**PHP SESSION** • Store data • Session and cookies • APIs'	• Learn how to use PHP session with web pages • Understand how data store to cookies • Learn how to secure database and website		PHP and MySQL Web Development (5th Edition) http://ebook-dl.com/book/3826	Hands On Programming Short Quiz
Week 15-16	**PROJECT CONSULTATION**	• Understand business process of selected project title • Learn the requirements of target user • Identify different free web hosting sites • Learn basic project management			Project Deliberation
Week 17	**WEBSITE PROJECT PRESENTATION**	• Culminating activity given to the students to test their mastery of the course by developing application programs utilizing all the theories and concepts acquired	Project Presentation System Walk-through Simulation	*Application Project Documentation* *Developed System*	Project Deliberation
Week 18	**FINAL EXAM**				Final Examination

Grading System:

How Web Assignments are Graded
- 35% - All directions are followed
- 30% - If it compiles without error
- 35% - If it runs to completion with proper input and output

Grading Midterm		Final
Lecture **70%**		**Lecture** **70%**
Class Standing 60 %		Class Standing 60 %
• Short Quizzes 35%		• Short Quizzes 35%
• Seatworks /Exercises 25%		• Seatworks /Exercises 25%
• Long Quizzes 45%		• Long Quizzes 45%
Mid-term Examination 40%		Final Examination 40%
Laboratory **30%**		**Laboratory** **30%**
Class Standing 60%		Class Standing 60%
• Laboratory Exercises 40%		•Laboratory Exercises 40%
• Project 35%		•Project 35%
• Practical Examination 25%		•Practical Examination 25%
Passing Mark: 75%		Passing Mark: 75%

References: Thomson, Laura and Welling, Luke. PHP and MySQL Web Development 5th edition, Addison-Wesley, 2016

Required Readings :
1. Anderson-Freed, Susan. **Weaving a Website**. Prentice Hall. 2002.
2. Larry Ullman, **PHP for World Wide Web** 3rd Edition. Peachpit Press 2004

Suggested Readings :
1. Lior, Linda Newman. Writing for Interaction: Crafting the Information Experience for Web and Software Apps. Elsevier, 2013.
2. Lengstorf. Real Time Web Apps. 2013.
3. Wang. Dynamic Web Programming and HTML5. 2013
4. Sklar, Joel. Web Design 2nd Edition. Cengage Learning, 2012
5. Vodnik, Sasha. HTML5 and CSS3 – Introductory, Cengage Learning, 2012
6. Shelley, Campbell. Web Design, Introductory. 2012.

General Rules :

1. The course is expected to have a minimum of three (3) quizzes. **No make up tests will be given**.
2. Assignments and research projects/report works will be given throughout the semester. Such requirements shall be due as announced in class. Late submission shall be penalized with grade deductions (5% per day) or shall no longer be accepted, depending on the subject facilitator's discretion. Assignments and exercises are designed to assist you in understanding the materials presented in class, and to prepare you for the exams.
3. Students are required to attend classes regularly, including possible make-up classes. **It is the student's responsibility to gather information missed due to absence.** The university guidelines on attendance and tardiness will be implemented.
4. **Any evidence of copying or cheating during any examinations may result in a failing grade from the examination for all parties involved. Note that other university guidelines shall be used in dealing with this matter**.

Prepared by:	Enhanced by:	Reviewed by:	Approved by:
COMMITTEE ON COMP 20163	**Prof. Cherry M. Doromal** **Facilitator**	**Prof. Doris B. Gatan** **Head, Academic Program**	**Prof. Edgardo S. Delmo** **Branch Director** **Dr.Emmanuel C. De Guzman** Vice President for Academic Affairs

Table of Contents

LESSON 1

EXPLORING THE WEB

INTRODUCTION

This lesson will be a brief history of the development of the Internet along with current trends and emerging technologies such as virtualization, botnets, internet of things, and more.

LEARNING OUTCOMES

At the end of this module, the students are expected to:

1. Know the brief history of the development of the Internet and World Wide Web
2. Understand the role of web server, web browser, and client
3. Understand how the Internet works and the technology behind it.
4. Familiarize student different websites, feature and technologies

COURSE CONTENTS

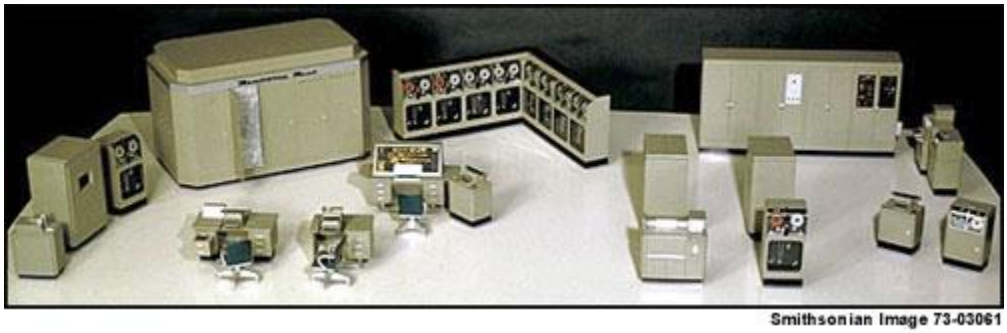
UNIT 1 – Internet

Brief History of The Internet

The Internet started in the 1960s as a way for government researchers to share information. Computers in the '60s were large and immobile and in order to make use of information stored in any one computer, one had to either travel to the site of the computer or have magnetic computer tapes sent through the conventional postal system.

Another catalyst in the formation of the Internet was the heating up of the Cold War. The Soviet Union's launch of the Sputnik satellite spurred the U.S. Defense Department to consider ways information could still be disseminated even after a nuclear attack. This eventually led to the formation of the ARPANET (Advanced Research Projects Agency Network), the network that ultimately evolved into what we now know as the Internet. ARPANET was a great success but membership was limited to certain academic and research organizations who had contracts with the Defense Department. In response to this, other networks were created to provide information sharing.

January 1, 1983 is considered the official birthday of the Internet. Prior to this, the various computer networks did not have a standard way to communicate with each other. A new communications protocol was established called Transfer Control Protocol/Internetwork Protocol (TCP/IP). This allowed different kinds of computers on different networks to "talk" to each other. ARPANET and the Defense Data Network officially changed to the TCP/IP standard on January 1, 1983, hence the birth of the Internet. All networks could now be connected by a universal language.



Smithsonian Image 73-03061

The image above is a scale model of the UNIVAC I (the name stood for Universal Automatic Computer) which was delivered to the Census Bureau in 1951. It weighed some 16,000 pounds, used 5,000 vacuum tubes, and could perform about 1,000 calculations per second. It was the first American commercial computer, as well as the first computer designed for business use. (Business computers like the UNIVAC processed data more slowly than the IAS-type machines, but were designed for fast input and output.) The first few sales were to government agencies, the A.C. Nielsen Company, and the Prudential Insurance Company. The first UNIVAC for business applications was installed at the General Electric Appliance Division, to do payroll, in 1954. By 1957 Remington-Rand (which had purchased the Eckert-Mauchly Computer Corporation in 1950) had sold forty-six machines.

A Brief History of the Web

The Web was born in a particle physics laboratory (CERN) in Geneva, Switzerland in 1989. There a computer specialist named Tim Berners-Lee first proposed a system of information management that used a “hypertext” process to link related documents over a network. He and his partner, Robert Cailliau, created a prototype and released it for review. For the first several years, web pages were text-only. It’s difficult to believe that in 1992, the world had only about 50 web servers, total. The real boost to the Web’s popularity came in 1992 when the first graphical browser (NCSA Mosaic) was introduced, and the Web broke out of the realm of scientific research into mass media. The ongoing development of web technologies is overseen by the World Wide Web Consortium (W3C).

Milestone of the Web

- Internet (1960s)
- World Wide Web - WWW (1991)
- First Web Browser - Netscape, 1994
- Google, 1998
- Facebook, 2004
- Smartphones (iPhone), 2007
- Tablets (iPad), 2010

The Internet Versus the Web

The *Internet* is a network of connected computers. No company owns the Internet; it is a cooperative effort governed by a system of standards and rules. The purpose of connecting computers together, of course, is to share information. There are many ways information can be passed between computers, including email, file transfer (FTP), and many more specialized modes upon which the Internet is built. These standardized methods for transferring data or documents over a network are known as protocols.

The *Web* (originally called the World Wide Web, thus the "www" in site addresses) is just one of the ways information can be shared over the Internet. It is unique in that it allows documents to be linked to one another using *hypertext* links—thus forming a huge "web" of connected information. The Web uses a protocol called *HTTP (HyperText Transfer Protocol).* That acronym should look familiar because it is the first four letters of nearly all website addresses, as we'll discuss in an upcoming section.

The Web is a subset of the Internet. It is just one of many ways information can be transferred over networked computers.

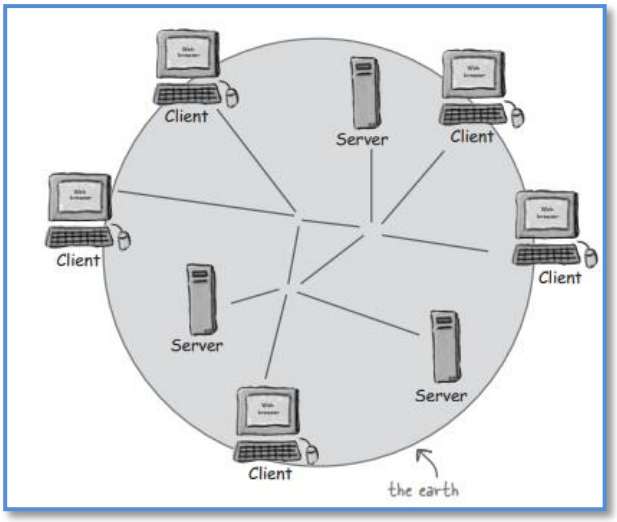
Intranets and Extranets

When you think of a website, you generally assume that it is accessible to anyone surfing the Web. However, many companies take advantage of the awesome information sharing and gathering power of websites to exchange information just within their own business. These special web-based networks are called *intranets*. They are created and function like ordinary websites, but they use special security devices (called firewalls) that prevent the outside world from seeing them. Intranets have lots of uses, such as sharing human resource information or providing access to inventory databases.

An *extranet* is like an intranet, only it allows access to select users outside of the company. For instance, a manufacturing company may provide its customers with passwords that allow them to check the status of their orders in the company's orders database. Of course, the passwords determine which slice of the company's information is accessible.

UNIT 2 – How Things Woks on Web

The *web* consists of gazillions of clients using browsers and servers connected thru wires and wireless networks



Web Application

A web application is an application that is accessed with a web browser over the Internet or an intranet.

Web Browser

A *web browser* lets a user to request a resource. It is a program you use to access content on the web. There are a variety of browsers available but the most popular are:

- Edge (from Microsoft)
- Firefox (from Mozilla)
- Safari (from Apple)
- Chrome (from Google)
- Opera (Opera Software)

They all provide similar functionality so if you're going to be using the basic functionality you can really use of any of these browsers. All of the above browsers have all the necessary functionality to support you browsing the web

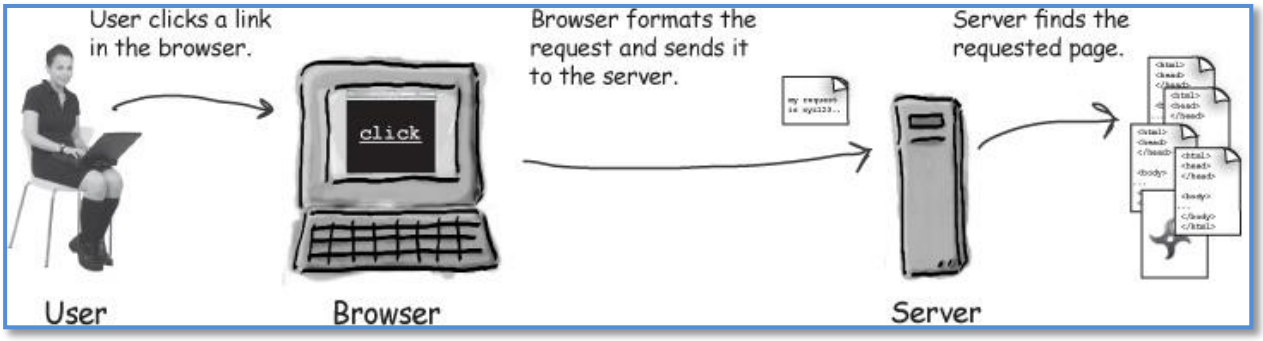
Web Client

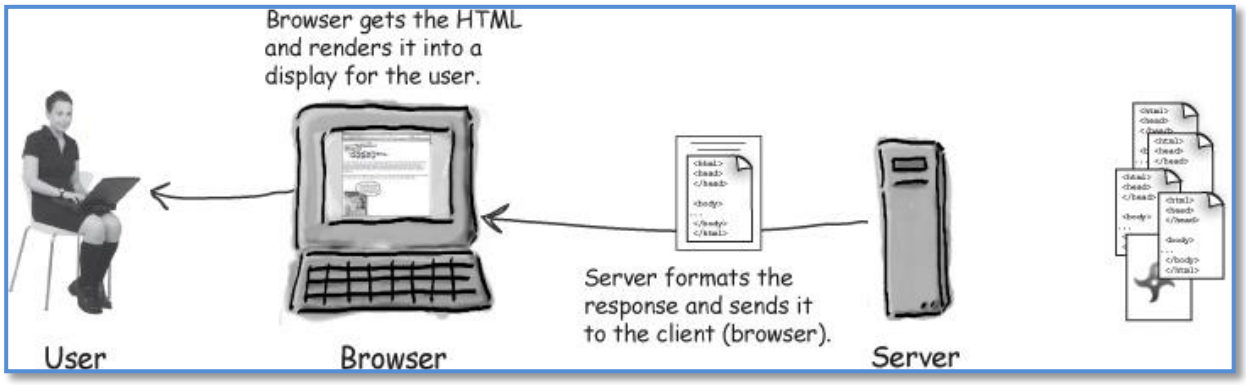
A *web client* lets the user request something on the server and shows the user the result of the request.

When we talk about *clients*, though, we usually mean both (or either) the human user and the browser *application*.

The *browser* is the piece of software (like Netscape or Mozilla) that knows how to communicate with the server. The browser's other big job is interpreting the HTML code and rendering the web page for the user.

When we use the term client, we usually won't care whether we're talking about the human user or the browser app. In other words, the client is the browser app doing what the user asked it to do.





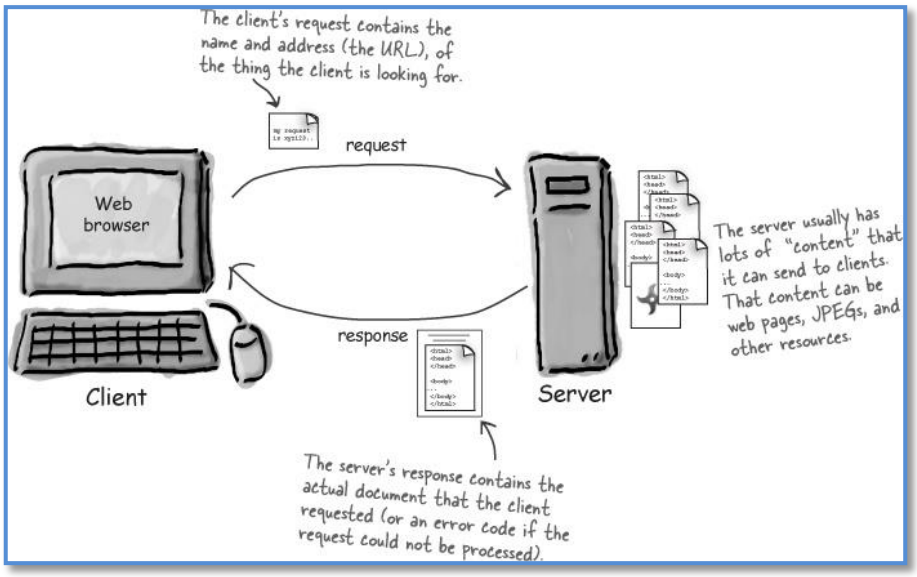
Web Server

A *web server* takes a client request and gives something back to the client.

A *web* browser lets a user request a resource. The web server gets the request, finds the resource, and returns something to the user. Sometimes that resource is an *HTML page*. Sometimes it's a picture. Or a sound file. Or even a PDF document. Doesn't matter—the client asks for the thing (resource) and the server sends it back.

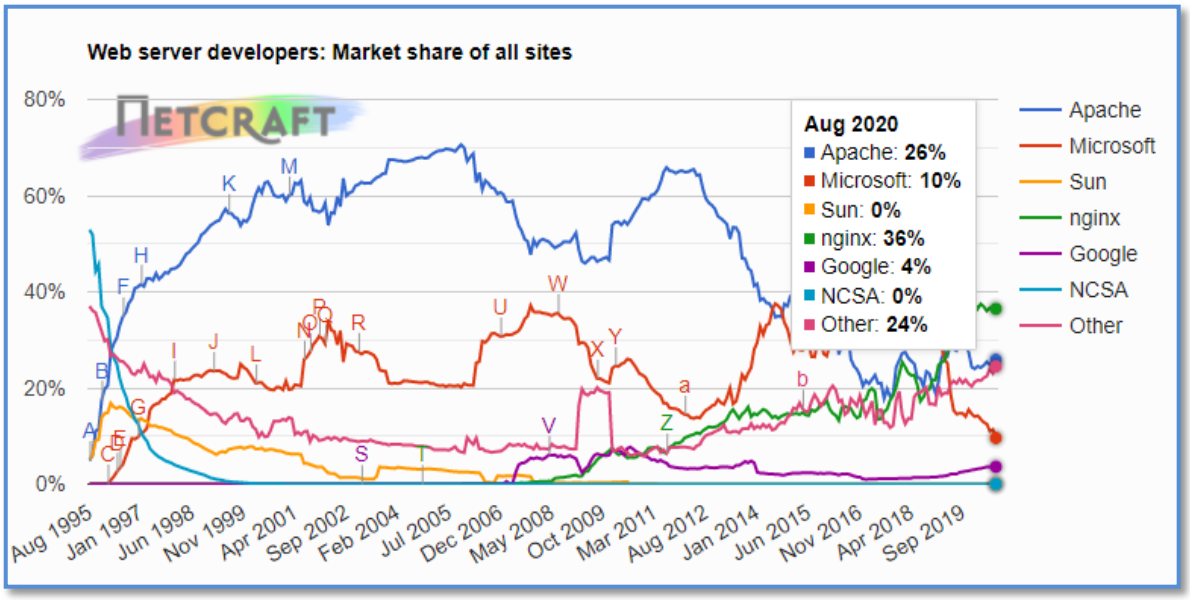
Unless the thing isn't there. Or at least it's not where the server is expecting it to be. You're of course quite familiar with the "404 Not Found" error—the response you get when the server can't find what it thinks you asked for.

When we say "server", we mean either the physical machine (hardware) or the web server application (software) that helps to deliver web content that can be accessed through the Internet



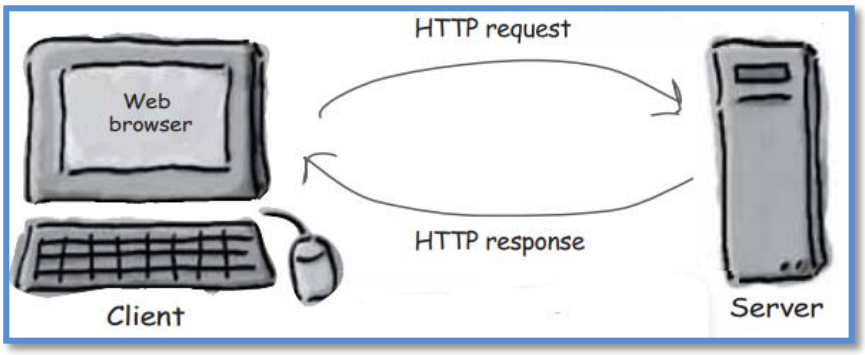
The most common use of web servers is to host websites

- **IIS** - Internet Information Services
  - Microsoft Windows
- **Apache** Web Server
  - Open Source
  - Cross-platform: UNIX, Linux, OS X, Windows
- **Nginx** (pronounced "engine x") - Has become very popular lately
- GWS (Google Web Server)



HyperText Transfer Protocol (HTTP)

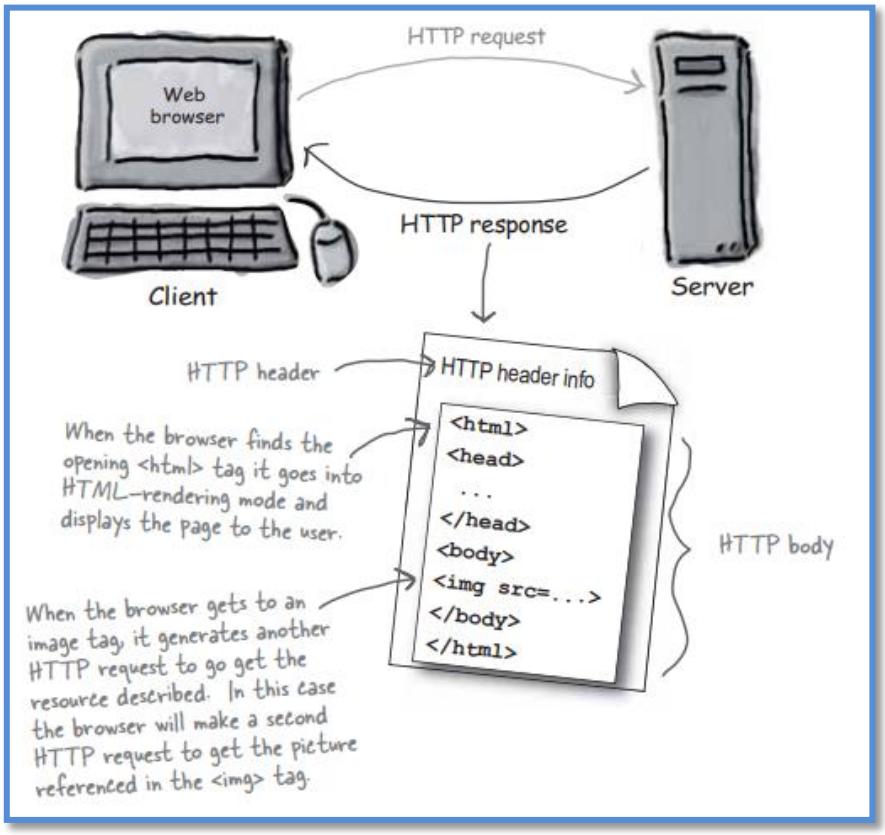
HyperText Transfer Protocol (HTTP) is the underlying protocol used by the World Wide Web to define how messages are formatted and transmitted and what actions Web servers and browsers should take in response to various commands. It is a request-response protocol in the client-server computing model. *Clients* and *servers* communicate by exchanging individual messages. The message sent by the client, typically a Web *browser*, is the request while the message sent by the server as an answer is the response. The server will provide resources such as *HTML* files, which is which contain the information for formatting and displaying Web pages.



It takes client request to server and it brings server response to client. Request/Response sequence; a browser requests and a server responds.

HyperText Markup Language (HTML)

Web pages use a markup language called *HyperText Markup Language*, or *HTML* for short, which was created especially for documents with hypertext links. Gives instructions to the browser. HTML defines dozens of text elements that make up documents such as headings, paragraphs, emphasized text, and of course, links. An HTTP response contain HTML.



Putting it All Together

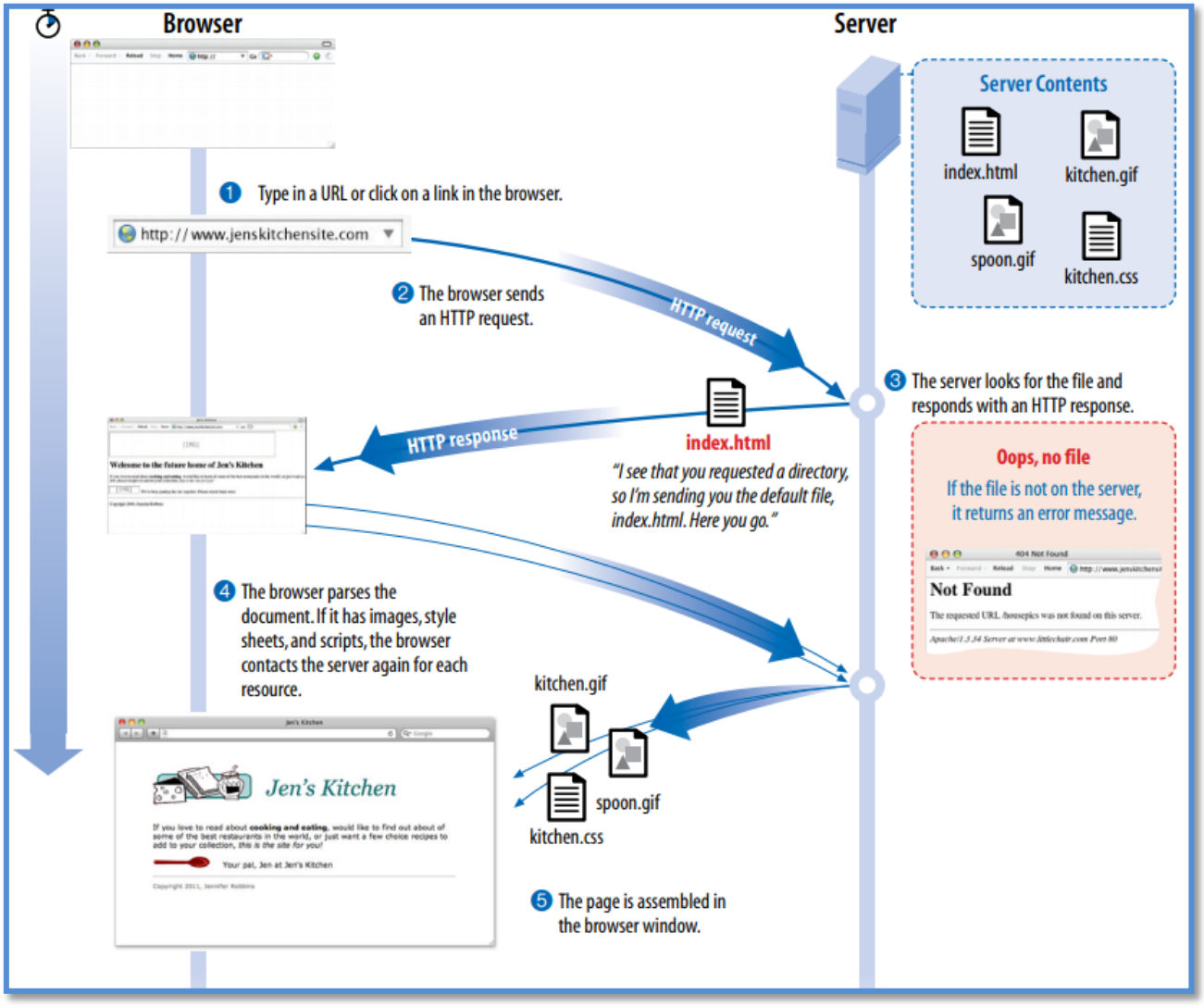
1. Client request a web page by either typing its URL directly in the browser or by clicking on a link on a page. The URL contains all the information needed to target a specific document on a specific web server on the Internet.
2. Browser sends an HTTP Request to the server named in the URL and asks for the specific file. If the URL specifies a directory (not a file), it is the same as requesting the default file in that directory.
3. The server looks for the requested file and issues an HTTP response.

a. If the page cannot be found, the server returns an error message. The message typically says “404 Not Found,” although more hospitable error messages may be provided.

b. If the document is found, the server retrieves the requested file and returns it to the browser.

4. The browser parses the HTML document. If the page contains images (or other external resources like scripts, the browser contacts the server again to request each resource specified in the markup.

5. The browser inserts each image in the document flow where indicated by the img element. The assembled web page is displayed for your viewing pleasure.



UNIT 3 –Web Technologies

As important as it is to know how we reached where we are today, it is also important to stay current in web development. New products and innovations can greatly affect the landscape in a short amount of time.

Cloud Computing

Cloud computing technology is increasingly being used by enterprises and organizations. It basically involves a variety of independent technologies such as hardware virtualization, distributed processing, utility computing, network system, web services, platform as a service, and software as a service.

The term *'cloud computing'* also refers to the technology that makes cloud work. This includes some form of *virtualized IT infrastructure*—servers, operating system software, networking, and other infrastructure that's abstracted, using special software, so that it can be pooled and divided irrespective of physical hardware boundaries.

The following are the Cloud Computing Services:

- SaaS (Software-as-a-Service)
- PaaS (Platform-as-a-Service)
- IaaS (Infrastructure-as-a-Service)

SaaS (Software-as-a-Service)

SaaS—also known as cloud-based software or cloud applications—is application software that's hosted in the cloud and that you access and use via a web browser, a dedicated desktop client, or an API that integrates with your desktop or mobile operating system

With *SaaS*, you take advantage of new features as soon as the provider adds them, without having to orchestrate an on-premises upgrade. Because your application data is in the cloud, with the application, you don't lose data if your device crashes or breaks.

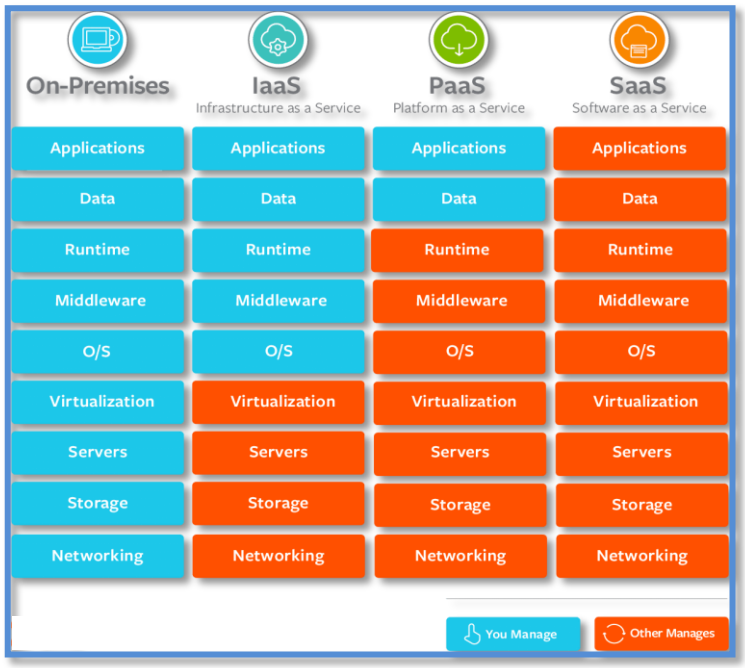
PaaS (Platform-as-a-Service)

PaaS provides software developers with on-demand platform—hardware, complete software stack, infrastructure, and even development tools—for running, developing, and managing applications without the cost, complexity, and inflexibility of maintaining that platform on-premises.

With *PaaS*, the cloud provider hosts everything—servers, networks, storage, operating system software, middleware, databases—at their data center. Developers simply pick from a menu to 'spin up' servers and environments they need to run, build, test, deploy, maintain, update, and scale applications.

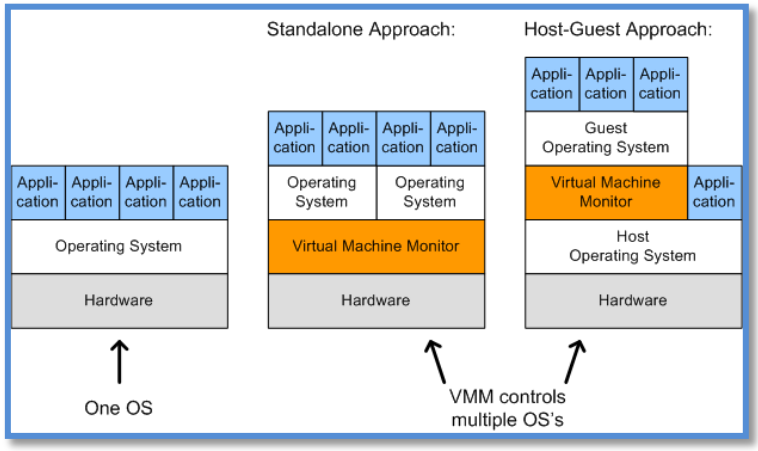
IaaS (Infrastructure-as-a-Service)

IaaS provides on-demand access to fundamental computing resources–physical and virtual servers, networking, and storage—over the internet on a pay-as-you-go basis. IaaS enables end users to scale and shrink resources on an as-needed basis, reducing the need for high, up-front capital expenditures or unnecessary on-premises or 'owned' infrastructure and for overbuying resources to accommodate periodic spikes in usage.



Virtualization

Server virtualization is the act of running multiple operating systems and other software on the same physical hardware at the same time. A hardware and/or software element is responsible for managing the physical system resources at a layer in between each of the operating systems and the hardware itself. Doing so allows the consolidation of physical equipment into fewer devices, and is most beneficial when the servers sharing the hardware are unlikely to demand resources at the same time, or when the hardware is powerful enough to serve all of the installations simultaneously.



Botnets

Botnets are not exactly a new threat to the Internet, but they remain one of the most persistent threats to the average user and their computer. The word *botnet*, an amalgamation of the words robot and network, is an accurate description of what it entails. *Botnets* are programs that use a network connection to communicate with each other to coordinate and perform tasks.

Some *botnet* controllers have grown so large and organized that they act as businesses in competition, typically "renting" their botnet out as a service or tool to others for agreed upon rates.. Simply shutting down or attempting to remove the malicious files from infected systems could cause unintended damage to the machines, further complicating the process of eliminating a botnet.

How a botnet works:

1. A botnet operator sends out viruses or worms, infecting ordinary users' computers, whose payload is a malicious application — the bot.
2. The bot on the infected PC logs into a particular command and control (C&C) server (often an IRC server, but, in some cases a web server).
3. A spammer purchases access to the botnet from the operator.
4. The spammer sends instructions via the IRC server to the infected PCs, causing them to send out spam messages to mail servers.

Internet of Things (IoT)

The *Internet of Things* is the concept of connecting any device (so long as it has an on/off switch) to the Internet and to other connected devices. The *IoT* is a giant network of connected things and people – all of which collect and share data about the way they are used and about the environment around them.

That includes an extraordinary number of objects of all shapes and sizes – from smart microwaves, which automatically cook your food for the right length of time, to self-driving cars, whose complex sensors detect objects in their path, to wearable fitness devices that measure your heart rate and the number of steps you've taken that day, then use that information to suggest exercise plans tailored to you. There are even connected footballs that can track how far and fast they are thrown and record those statistics via an app for future training purposes.

The *Internet of Things* is the concept of connecting any device (so long as it has an on/off switch) to the Internet and to other connected devices. The *IoT* is a giant network of connected things and people – all of which collect and share data about the way they are used and about the environment around them.

That includes an extraordinary number of objects of all shapes and sizes – from smart microwaves, which automatically cook your food for the right length of time, to self-driving cars, whose complex sensors detect objects in their path, to wearable fitness devices that measure your heart rate and the number of steps you've taken that day, then use that information to suggest exercise plans tailored to you. There are even connected footballs that can track how far and fast they are thrown and record those statistics via an app for future training purposes.

UNIT 4 –Web Design

Website design is a topic of study often neglected until after a programming background has been developed. A number of factors affect design in web development, complicating what would otherwise appear to the end user to be a relatively simple process of displaying a picture or document.

Planning Cycle

The planning process described is intended to build upon itself to refine project requirements, look and feel, and development plans. Starting early with programming during design planning can accelerate a project when the elements created early on are unlikely to be affected by later changes in the scope. It is important to avoid aspects that are assumed to change, like visual layout or particular pieces of content, instead focusing on data structure, frameworks, and other components that are easily adapted to design changes.

When considering a milestones, tasks, objectives, or whatever label you or your team place on objectives, a handy acronym to reference is *SMART*. *SMART* stands for Specific, Measurable, Attainable, Realistic, and Timely. By ensuring all of your objectives meet the *SMART* criteria, you will have a better chance of keeping your project on time and well planned.

Specific: Is your objective specific enough to convey its full scope? While you do not want to specify implementation of the objective, you should convey enough specific information that the person assigned to the objective can begin their portion of implementation.

Good Example: Deliver our standard proposal with adjusted price quotes to reflect customer's discount rate of 15%.

Bad Example: Deliver a proposal to the customer.

Measurable: Your objective should have a clear indicator of when it is complete.

Good Example: Complete the first 15 pages identified in the site plan.

Bad Example: Complete the first 20% of the site.

Attainable: Is it possible, at all, to complete the objective?

Good Example: Get the server to a FedEx store by close of business on delivery date.

Bad Example: Drive the server from Baguio to Manila within 24 hours.

Realistic: Is it possible to complete the objective given the timeline and resources on hand?

Good Example: Have Team A (staff of 20) complete 10 pages by tomorrow

Bad Example: Have Bob the Intern complete 10 pages by tomorrow.

Timely: Will the objective be useful if it is completed at (not near or before) its deadline?

Good Example: Have draft sitemap completed by the end of Wednesday to include in the proposal.

Bad Example: Have draft sitemap completed by the end of Friday to include in the proposal

The Fold

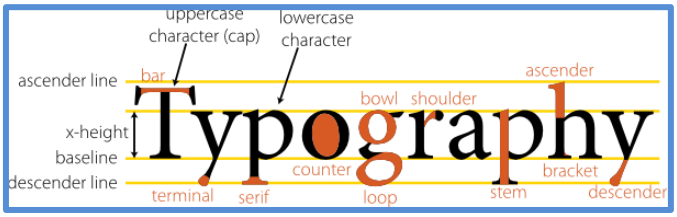
We need to consider where we want to place pieces of our content. If you look at newspapers, you will find that the most attractive story of the day (as decided, at least, by the publisher) is emblazoned in large letters near the middle or top of the front page, surrounded by the name of the paper, the date, and other pieces of information that quickly lend to your decision of whether or not to purchase a given paper. This is done intentionally, to make the paper attract your attention and get you to buy their edition over their competitors.

What you typically find here is the name and or logo of the company, and what they feel is most important for you to see first. As you begin to analyze web pages in this light, you will find it very easy to determine what kind of site they are, or what they want or expect from you as their guest. Companies will lead with a featured product or sale to attract your attention, and search engines will make the search bar prominent, usually with ad space close by to increase their revenue streams.

Typography

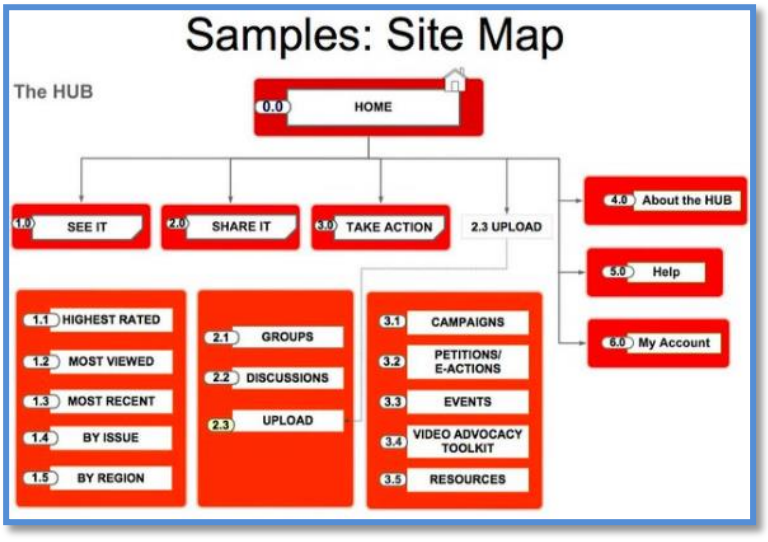
Typography is the study of font. Now, advances in CSS allow us to use unusual fonts by connecting to them through our styling. This allows us to use a tremendous variety of fonts in our sites to add to our look and feel, adding an aspect that has unlocked new approaches to design. Some of the elements of typography include the study of features like readability, conveying meaning or emotion through impression, and the artistic effect of mixing styles.

For ease of reading and to avoid a cluttered appearance, most sites keep to two or three fonts when creating their design. One for text, and one or two for headings, titles, and distinguishing marks. All of these should be kept in the same family for a more congruous experience, and each unusual font defined in your site should include fallback definitions in case there are problems loading your primary style



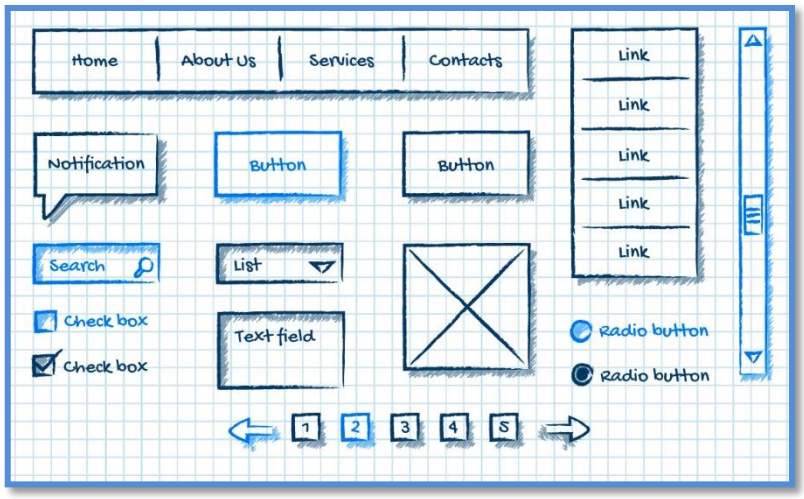
Site Maps

A *site map* is a file that contains a master list of links to the pages on your site, and can provide information about those pages like how often they are updated, how pages connect to each other, and how important it is relative to the other pages. It can be a reference tool to both Bots that index your site for search engines, as well as your visitors trying to find particular content. Site map files are XML documents arranged in hierarchical format that bots read to gain understanding of your site layout, page relevance, and organization. The file may also be a human readable page that diagrams how pages relate to one another, and serve as a master list of the pages in your site. Site maps are best kept in the root of your website, at the same level as your initial index page.



Wireframes

A *wireframe* may include things like location and size of elements such as a login button, where banners and content sections will sit, and provide an overall idea of how a site will operate. When wire framing a website, the idea is to create a mockup of one or more designs that portray how the interface might appear to the user. By the end of your wire framing process, you should have an idea of how the site will operate, and have resolved questions over where users will find particular features and elements.



Color Schemes

The process of determining the color(s) involved in your site could fill a book. In fact, it does. Regardless of the varying opinions of what emotions colors instill, or represent, the quickest way to alienate a user is to give them a visual experience that is unappealing. The layout, appearance, and cohesiveness of your site are something that are immediately judged when a user first visits. These elements influence everything from their impression of what the site represents, its reputability, and even its trustworthiness as an ecommerce option. If your site appears to be disorganized, dated and out of style, or seems too "busy" or complicated, you can lose users in less than ten seconds.

UNIT 5 –Web Design Development Tools

Frameworks

Frameworks are compilations of code that you can use to quickly start a site with a collection of features already in place. A typical framework is a set of files that come with their own instructions, and can be so extensive that they take on a life and syntax beyond the language they are written in. They extend the features normally found in the language they are written in by adding their own classes, functions, and capabilities. The goal is that by giving the framework a few commands, we can create much larger processes like a menu system or complete color scheme. Some frameworks focus on the automation of repetitive tasks like generating forms and pages based on tables in a database, or applying in depth style and structure across a website

Yii Framework

The Yii framework focuses on reducing SQL statement writing, follows Model View Controller methods, and helps create themes, web services, and integration with other platforms. It also includes security, authentication, and unit-based approaches to development.

Ruby on Rails

Quite popularly referred to, as RoR, Ruby on Rails has emerged as one of the favorites among the web developers today. Since its launch in 2005, RoR is still completely free to use, is open source, and runs on Linux. It is fun to work with and is remarkably quick in getting you through the planning stage and on to the developmental stage.

Angular JS

One of the most popular and well-known frameworks – Angular.js comes from the digital giant of the day, Google. It is essentially a JavaScript open-source framework that can help you make single web-page applications using an MVC (Model-Controller-View) architectural pattern. It is not really a full stack framework, but you can consider it as a front-end framework which is great for handling your WebPages. If you are building an e-commerce site, it is important that you have Angular.js in your toolkit for great results.

Node.js

Node.js is decidedly the best when you are looking to build a lightweight, yet highly efficient website. It is uncanny how perfectly well it works with real-time applications that have a huge amount of data running through on distributed devices. Node.js is more than a mere framework. It has the ability to ensure scalability and fast network applications as they are capable of dealing with multiple applications at the same time without compromising on performance.

Laravel

An open-source PHP based framework, Laravel is built with the intent of creating only the best quality applications. The framework offers a system that is equipped with a dedicated dependency manager, has an inclination towards syntactic sugar, and helps in the application maintenance and deployment. This, is precisely why Laravel is considered the best PHP based framework and is considered one of the most important frameworks in building an e-commerce website.

Django

Django is a popular framework when it comes to building quality web apps. The framework was created with the intent of meeting rapidly moving newsroom deadlines while ensuring that they still live up to the demanding requirements of seasoned web developers. Developers favor Django because they find it impressively quick, secure, scalable, and versatile. The language used in the framework is Python and the websites that have been developed using this framework include Disqus, Pinterest, Instagram, and Quora.

CodeIgniter

Created by EllisLab, CodeIgniter is a well-known web application framework for building dynamic websites. Loosely based on the MVC architecture, the framework mandates Controller classes, but models and views are optional here. The platform is quite popular as it promises and delivers exceptional performance, almost zero configuration and massive monolithic libraries. The language used in the framework is PHP and the websites built using this framework include Bufferapp, The Mail and Guardian

Zend Framework

Zend is an object-oriented web framework that is based on agile methodology and was built for development of enterprise-level applications. The framework is known for being characteristically fast, secure, and extendable. It means that this unique web development framework is open for customization. Developers who are planning to look at including some project-specific functions with minimum fuss find Zend to be perfect for them, as the framework adheres to PHP best practices.

Templates

Similar to the idea behind frameworks, templates are sets of files that dictate the basic structure that provides a layout to your site. Templates typically create a grid format you can select from, like two or three columns, fixed or relative width and height, etc. If you are starting a site fresh and putting it into an empty template, there may be some placeholder content and styling as well. Templates are useful for getting the look and feel of a site up and running fast and there is little concern about the particulars of appearance, or whenever the template meets your needs well.

Bootstrap

Bootstrap was created by Twitter in order to help them manage their extremely popular service. As it matured, they made it open source to allow others to utilize the toolset. Their framework provides tools for styling, interaction elements like forms and buttons, and navigation elements including drop downs, alert boxes, and more. Using this frame work involves little more than linking to the appropriate JavaScript and CSS files and then referencing the appropriate style classes in your code.

Foundation

The Foundation system focuses on front end design and follows the principles of responsive web design. Their approach uses a grid layout to allow flexibility, accelerate prototyping and wire framing, and provides integrated support for multi-platform designs.

Integrated Development Environments

Editing HTML and CSS code can be done with nothing but a simple text editor. However, if you'd like to take your programming skills (and output) to the next level, it's worth looking into an integrated development environment or IDE.

A bare-bones IDE will allow you to code, edit, test, and debug. However, an advanced IDE, like the ones listed here, will offer many features that will enhance your programming experiences like automation, visualization, and customization.

Visual Studio Code

Visual Studio Code is a powerful source code editor that comes with a range of tools for JavaScript development. The IDE comes with built-in support for JavaScript, TypeScript, and Node.js. It also has plenty of extensions for other languages (such as C++, C#, Python, and PHP). Developed by Windows, Visual Studio Code is great for new programmers as it explains everything from HTML tags to syntax and error handling.

Sublime Text

Sublime Text 3 is a flexible, free IDE for Windows, Mac, and Linux. It supports a number of different programming and markup languages, including Python, C, HTML, JavaScript, and CSS. The interface is known to be clutter-free and fast.

PyCharm

PyCharm is a decent free IDE for web development in a number of languages, including Python, CSS, HTML, JavaScript, Node.js, and more. The IDE is compatible with Mac, Windows, and Linux and has a paid sister software you can purchase if you'd like something more reliable. According to some users, the free version of PyCharm can be buggy, especially the autocomplete feature.

NetBeans

NetBeans is a free, easy-to-use IDE that works well with JavaScript, HTML, PHP, C, and C++. As well as supporting a number of different programming languages, it also is available in English, Brazillian Portuguese, Japanese, Russian, and Simplified Chinese. NetBeans isn't ideal for those just starting programming, as the package can be tricky to set up.

Bluefish

Bluefish is a cross-platform, lightweight IDE that can be used with Windows, macOS, Solaris, and many Linux distros. It supports a variety of different programming languages, including HTML, CSS, Perl, SQL, Ruby, PHP, Python, and more.

Notepad ++

Notepad++ is a source code text editor with syntax highlighting, multiple document handling using tabs, auto-completion of keywords (customizable), regular expressions in the search and replace

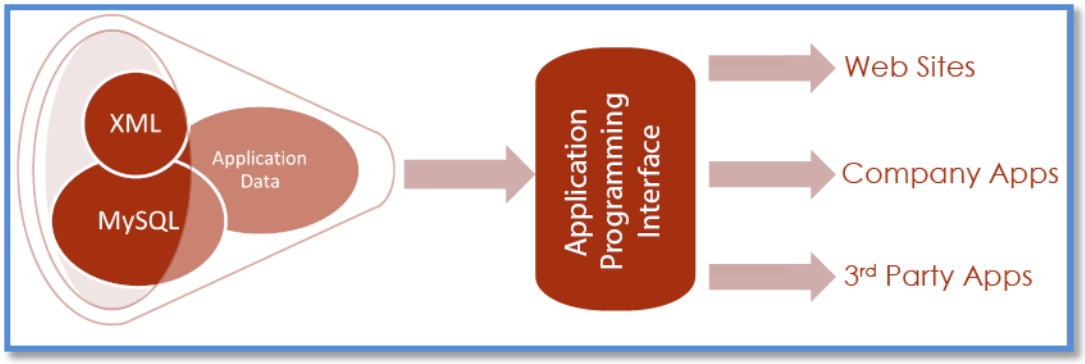
function, macro recording and playback, brace and indent highlighting, collapsing and expanding of sections of code, and more.

Application Programming Interfaces

Commonly referred to as APIs, pronounced as the letters of the acronym, application programming interfaces allow us to interact with features and data of a system other than its primary means, whether it is an application or website. Created to address needs of data exchange and integration between systems, APIs provide a controlled method of allowing others to use a system without having direct, unfettered access to the code or database it runs on.

Web-based APIs are, essentially, limited websites. They allow the pages and scripts end users create to communicate with the data source by using a predetermined vocabulary and fixed amount of options. When the user's message reaches the API, the API completes the requested task such as getting a certain piece of data, or validating credentials, and returns the results, hiding anything the developers do not want revealed, and only provides the features they are comfortable with others using

By combining several disparate systems through their APIs, a mashup is created, which is a new feature, application, or service created by combing several others. APIs are often included as part of a software development kit (SDK) that includes the API, programming tools, and documentation to assist developers in creating applications.



ASSESSMENT/ACTIVITIES GUIDE QUESTION/s

Guide Questions

1. Explain how things work on web.
2. What is the difference between the world wide web and the internet?
3. Describe the difference between APIs, frameworks, and templates.
4. Find and describe an example of the Internet of Things in use today
5. How botnets work?
6. Describe how a web server works, and the steps that take place from your initial request for a world wide web document using a web browser until the requested document renders in your browser. Be specific.
7. In the article “The Internet Under Siege”, Lawrence Lessig describes the Internet as being built with three layers. The physical layer, the code layer and the content layer. The code layer is a commons, the other two are generally not a commons. Describe how these three layers operate, and their impact on the development of the Internet, as well as Internet content.

LESSON 2

HTML

INTRODUCTION

Servers often send the browser a set of instructions written in HTML (HyperText Markup Language). It tells the browser how to present the content to the user. It is a language for describing web pages.

LEARNING OUTCOMES

At the end of this module, the students are expected to:

1. Familiarize with the HTML structure programming and tags and elements.
2. Understajd about markup, Headings and paragraphs, Bold, Italic, emphasis
3. Creating links between pages, linking to other sites and email link
4. Learn to create tables and what information suits to tables.
5. Understand different versions of HTML, elements, comments, meta information and iframes

COURSE CONTENTS

UNIT 1 – Markup Languages

Document markup is a notation method that defines how particular pieces of information are meant to be formatted. The term comes from the practice of marking up manuscripts to notate changes that need to be made. Markup in terms of programming languages is used to identify a language that specifies how a document is to appear.

HyperText Markup Language (HTML)

HTML is not a programming language, it is a markup language. A markup language is a set of markup tags. *HTML* uses markup tags to describe web pages.

Hypertext markup language is used to aid in the publication of web pages by providing a structure that defines elements like tables, forms, lists and headings, and identifies where different portions of our content begin and end. It can be used to embed other file formats like videos, audio files, documents like PDFs and spreadsheets, among others. HTML is the most relied upon language in the creation of web sites

An *HTML* file must have an htm or html file extension.

History

In the beginning, back to the first days of the Internet and ARPA, the primary purpose of creating a page was to share research and information. *HTML* tags were only meant to provide layout and formatting of a page. As such, early implementations of *HTML* were somewhat limited as there was little demand for features beyond the basics. Headings, bullets, tables, and color were about all developers had to utilize. As sites were created for other more commercial uses, developers found creative ways of using these tools to get their pages looking more like magazines, advertisements, and what they had drawn on paper.

HTML5

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and latest major version of HTML that is a *World Wide Web Consortium (W3C)* recommendation.

HTML5 was first released in public-facing form on 22 January 2008, with a major update and "*W3C* Recommendation" status in October 2014. Its goals were to improve the language with support for the latest multimedia and other new features; to keep the language both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, etc., without XHTML's rigidity; and to remain backward-compatible with older software

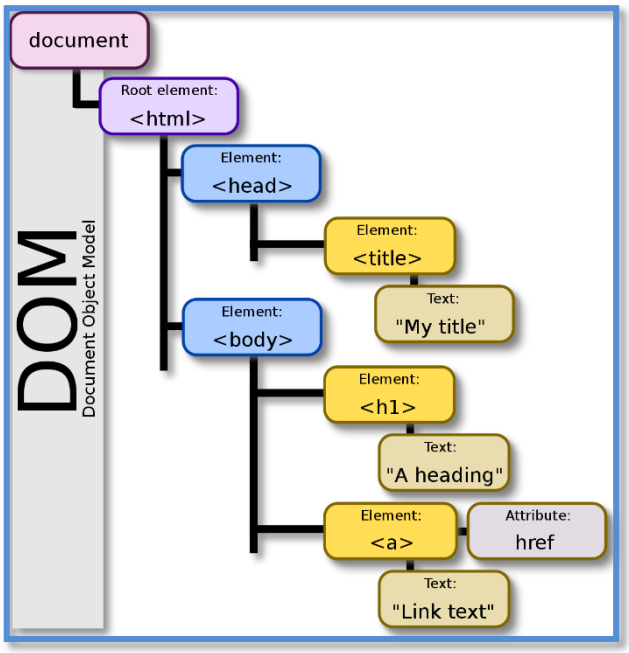
W3C Standards

The *World Wide Web Consortium*, or *W3C*, is an international community that supports web development through the creation of open standards that provide the best user experience possible for the widest audience of users. This group of professionals and experts come together to determine how CSS and HTML should operate, what tags should be included as features, and more. The *W3C* is also your best reference point in determining the accessibility of your site through the use of tools that analyze your code for *W3C* compliance. These tools confirm if you have fully implemented elements in your code, like providing alternate text descriptions of images in the event that the image cannot load, or the user is visually impaired.

Document Object Model

Document Object Model defines the order and structure of document files as well as how the file is manipulated to create, edit, or remove contents.

The *DOM* is built to be language and platform independent so any software or programming language can use it to interface with documents. It defines the interface methods and object types that represent elements of documents, the semantics and behavior of attributes of those objects, and also defines how they relate to one another. The *DOM*, effectively, is what gives rise to the tags we are about to study below.



The diagram is an example of a document's model in tree format, with nested elements appearing to the right and below their parents. In this example, we are shown an HTML page with a section for the head and the body, which includes a page title and a link as its contents. This structure provides the ability for us to traverse, or move around the document, by referring to an object's name or attribute

HTML Document Structure

An *HTML document i*s a file containing Hypertext Markup Language, and its filename most often ends in the . *html* extension.

An *HTML document* is a text document read in by a Web browser and then rendered on the screen.

HTML documents contain HTML tags and plain text. HTML documents are also called web pages

When an HTML document is loaded into a web browser, it becomes a document object.

HTML Tags

HTML markup tags are usually called *HTML tags.* Tags are used to mark the beginning and end of document content. They instruct web browsers as to the type of content they contain, so that the browsers will know how to render the document's content.

Tags are enclosed within < and > brackets. The opening bracket (<) identifies the beginning of the tag. It is followed by the tag name. Tag names end with a closing bracket (>)

Tags Pairs

Most *HTML tags* work in pairs, including a start and an end tag. The first tag in a pair is the start tag, the second tag is the end tag. Start and end tags are also called opening tags and closing tags. The slash (/) indicates an ending tag

Syntax:

```
<tag> content </tag>
```

In the syntax:

`<start tag>`	identifies the beginning of the element
`content`	represents the content that is embedded within the two tags.
`</end tag>`	identifies where the elements ends

Single Tags

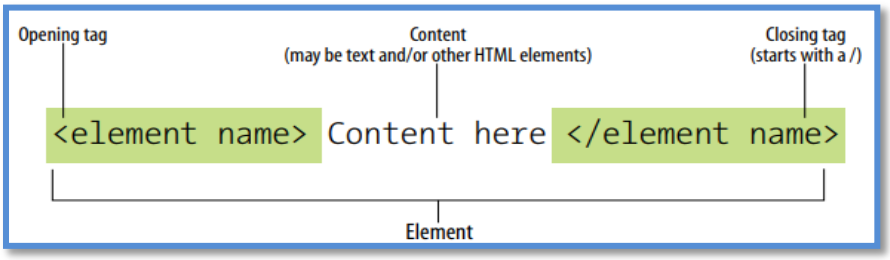
Elements that do not have an end tag are referred to as single or empty tags. *Single tags* do not contain any contents. All *single tags* are self-closed, which is accomplished by placing a / before the closing >

Syntax:

```
<tag/>
```

HTML Elements

An *HTML element* consists of both the content and its markup

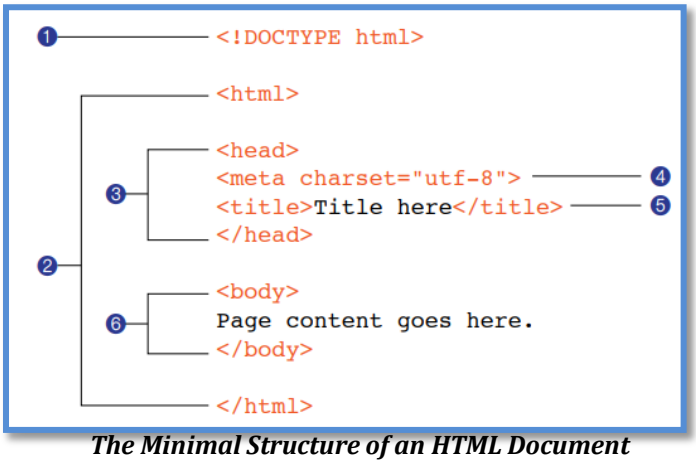


The parts of an HTML container element

Elements are identified by tags in the text source. A tag consists of the element name (usually an abbreviation of a longer descriptive name) within angle brackets (< >). The browser knows that any text within brackets is hidden and not displayed in the browser window.

Elements	Definition
body	Identifies the body of the document that holds the content
head	Identifies the head of the document that contains information about the document
html	The root element that contains all the other elements
meta	Provides information about the document
title	Gives the page a title

Parts of HTML Document



The Minimal Structure of an HTML Document

1. ***DOCTYPE declaration***
   Document type declaration is an instruction to the web browser about that this document is an HTML5 document.

2. ***<html >***
   The entire document is contained within an *html* element. The html element is called the root element because it contains all the elements in the document, and it may not be contained within any other element. It is used for both HTML and XHTML documents

3. ***<head>***
   Within the *html* element, the document is divided into a *head* and a *body*. The head element contains descriptive information about the document itself, such as its title, the style sheet(s) it uses, scripts, and other types of "meta" information

4. ***<meta>***

   The ***meta*** elements within the head element provide information about the document itself. A *meta* element can be used to provide all sorts of information, but in this case, it specifies the character encoding (the standardized collection of letters, numbers, and symbols) used in the document.

5. ***<title>***

   Also in the head is the mandatory *title* element. According to the HTML specification, every document must contain a descriptive title.

6. ***<body>***

   The *body* element contains everything that we want to show up in the browser window.

Adding Elements To The Head Section

<style> This tag is used to embed an internal style sheet into a document

```
<head>
<style type = "text/css">
        h1 {color:red}
        p {color:blue}
</style>
</head>
```

<link> This tag is used to set up a link to an external style sheet document

```
<head>
        <link rel="stylesheet" type="text/css" href="test.css" />
</head>
```

test.css

```
h1 {color:red;}
```

<script> This tag is used to add scripts to your documents

```
<head>
        <script language="javascript" type="text/javascript">
            document.write("Hello World!");
        </script>
</head>
```

Building Basic HTML

HTML

```
<html>
   <head>
        <title>Hello, World</title>
   </head>
   <body>
        <h1>Welcome!</h1>
         <p> Join us in learning the world of
           <em>web pages</em>.
         </p>
        <h2>Attention</h2>
        <p> Please </p>
  </body>
</html>
```

RESULT

Welcome!

Join us in learning the world of *web pages*.

Attention

Please

UNIT 2 – Text

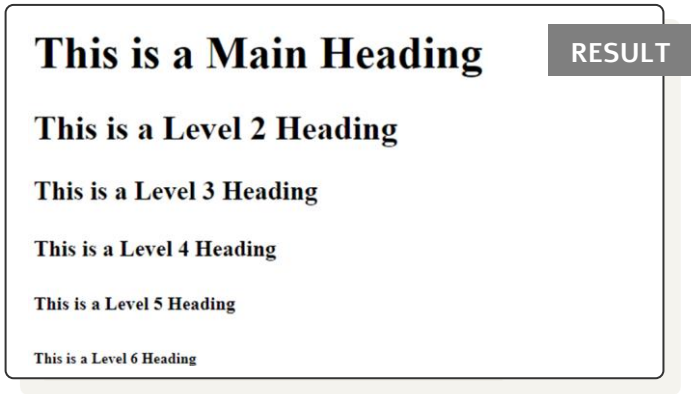
Headings

HTML has six "levels" of headings. Headings are defined with the <h1> to <h6>. <h1> defines the largest heading while <h6> defines the smallest. HTML automatically adds an extra blank line before and after a heading.

HTML

```
<h1>This is a Main Heading</h1>
<h2>This is a Level 2 Heading</h2>
<h3>This is a Level 3 Heading</h3>
<h4>This is a Level 4 Heading</h4>
<h5>This is a Level 5 Heading</h5>
<h6>This is a Level 6 Heading</h6>
```

RESULT



Paragraphs

Paragraphs are defined with the <p> tag. You indicate a paragraph with the p element by inserting an opening tag at the beginning of the paragraphh and a closing tag after it Think of a paragraph as a block of text. You can use the align attribute with a paragraph tag as well.

HTML

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

RESULT

This is a paragraph

this is another paragraph

Line Breaks

The
 tag is used when you want to start a new line, but don't want to start a new paragraph. The
 tag forces a line break wherever you place it. It is similar to single spacing in a document.

```
<p>The Earth<br />gets one hundred tons heavier
every day<br />due to falling space dust.</p>
```

HTML

The Earth
gets one hundred tons heavier every day
due to falling space dust.

RESULT

Horizontal Rules

The <hr> element is used for horizontal rules that act as dividers between sections

```
<p>Venus is the only planet that rotates
clockwise.</p> <hr />
<p>Jupiter is bigger than all the other planets
combined.</p>
```

HTML

Venus is the only planet that rotates clockwise.

Jupiter is bigger than all the other planets combined.

RESULT

Bold & Italic

By enclosing words in the tags and we can make characters appear bold.

By enclosing words in the tags and we can make characters appear italic.

```
<p>This is how we make a word appear
<b>bold.</b></p>
<p>This is how we make a word appear
<i>italic</i>.</p>
```

HTML

This is how we make a word appear **bold.**

This is how we make a word appear *italic*.

RESULT

Superscript & Subscript

The <sup> element is used to contain characters that should be superscript such as the suffixes of dates or mathematical concepts like raising a number to a power such as 2^2.

The <sub> element is used to contain characters that should be subscript. It is commonly used with foot notes or chemical formulas such as H_2 0.

HTML

```
<p>On the 4<sup>th</sup> of September you will learn about E=MC<sup>2</sup>.</p>
<p>The amount of CO<sub>2</sub> in the atmosphere grew by 2ppm in 2009<sub>1</sub>.</p>
```

RESULT

On the 4th of September you will learn about $E=MC^2$.

The amount of CO_2 in the atmosphere grew by 2ppm in 2009_1.

Strong & Emphasis

The use of the element indicates that its content has strong importance. By default, browsers will show the contents of a element in bold.

The element indicates emphasis that subtly changes the meaning of a sentence. By default browsers will show the contents of an element in italic.

HTML

```
<p><strong>Beware:</strong> Pickpockets operate in this area.</p>
<p>This toy has many small pieces and is <em>not suitable for children under five years old.</em></p>
```

RESULT

Beware: Pickpockets operate in this area.

This toy has many small pieces and is *not suitable for children under five years old.*

UNIT 3 – Lists

Ordered Lists

Ordered lists are lists where each item in the list is numbered. The ordered list is created with the <ol> element . Each item in the list is placed between an opening <li> tag and a closing tag. (The li stands for list item.)

HTML

```
<ol>
  <li>Chop potatoes into quarters</li>
  <li>Simmer in salted water for 15-20 minutes until tender</li>
  <li>Heat milk, butter and nutmeg</li>
  <li>Drain potatoes and mash</li>
  <li>Mix in the milk mixture</li>
</ol>
```

RESULT

1. Chop potatoes into quarters
2. Simmer in salted water for 15-20 minutes until tender
3. Heat milk, butter and nutmeg
4. Drain potatoes and mash
5. Mix in the milk mixture

Unordered Lists

Unordered lists are lists that begin with a bullet point (rather than characters that indicate order). The ordered list is created with the <ul> element . Each item in the list is placed between an opening <li> tag and a closing tag. (The li stands for list item.)

HTML

```
<ul>
    <li>1kg King Edward potatoes</li>
    <li>100ml milk</li>
    <li>50g salted butter</li>
    <li>Salt and pepper to taste</li>
</ul>
```

RESULT

- 1kg King Edward potatoes
- 100ml milk
- 50g salted butter
- Salt and pepper to taste

Definition Lists

Definition lists are made up of a set of terms along with the definitions for each of those terms. The definition list is created with the <dl> element and usually consists of a series of terms and their definitions. Inside the <dl> element you will usually see pairs of <dt> and <dd>elements.

<dt> is used to contain the term being defined (the definition term).

<dd> is used to contain the definition.

HTML

```
<dl>
  <dt>Sashimi</dt>
    <dd>Sliced raw fish that is served with condiments  wasabi and soy sauce
        </dd>
  <dt>Scale</dt>
    <dd>A device used to accurately measure the weight of
        ingredients</dd>
    <dd>A technique by which the scales are removed a fish</dd>
</dl>
```

RESULT

Sashimi
Sliced raw fish that is served with condiments wasabi and soy sauce
Scale
A device used to accurately measure the weight of ingredients
A technique by which the scales are removed a fish

Nested lists

You can put a second list inside an element to create a sublist or nested list.

HTML

```
<ul>
    <li>Mousses</li>
    <li>Pastries
        <ul>
            <li>Croissant</li>
            <li>Palmier</li>
            <li>Profiterole</li>
        </ul>
    </li>
    <li>Tarts</li>
</ul>
```

RESULT

- Mousses
- Pastries
  - Croissant
  - Palmier
  - Profiterole
- Tarts

UNIT 4 – Links

Links are the defining feature of the web because they allow you to move from one web page to another — enabling the very idea of browsing or surfing.

Writing Links

Links are created using the <a> element. Users can click on anything between the opening <a> tag and the closing tag. You specify which page you want to link to using the `href` attribute

.

The text between the opening `<a>` tag and closing `</a>` tag is known as link text. Where possible, your link text should explain where visitors will be taken if they click on it (rather than just saying "click here"). Below you can see the link to IMDB that was created on the previous page.

The `href` Attribute

The `href` (hypertext reference) attribute provides the address of the page or resource (its URL) to the browser. The URL must always appear in quotation marks. There are two ways to specify the URL:

- Absolute URLs provide the full URL for the document, including the protocol (http://), the domain name, and the pathname as necessary. You need to use an absolute URL when pointing to a document out on the Web (i.e., not on your own server).
  Example: `href`="http://www.oreilly.com/"
- Relative URLs describe the pathname to a file relative to the current document. Relative URLs can be used when you are linking to another document on your own site (i.e., on the same server). It doesn't require the protocol or domain name—just the pathname.
  Example: `href`="recipes/index.html"

Linking to Pages on the Web

Many times, you'll want to create a link to a page that you've found on the Web. This is known as an "external" link because it is going to a page outside of your own server or site. To make an external link, you need to provide the absolute URL, beginning with http:// (the protocol). This tells the browser, "Go out on the Web and get the following document."

Links are created using the element which has an attribute called `href`. The value of the `href` attribute is the page that you want people to go to when they click on the link. Users can click on anything that appears between the opening tag and the closing tag and will be taken to the page specified in the `href` attribute.

When you link to a different website, the value of the `href` attribute will be the full web address for the site, which is known as an absolute URL.

HTML

```
<p>Movie Reviews:
    <ul>
        <li><a href="http://www.empireonline.com"> Empire</a></li>
        <li><a href="http://www.metacritic.com"> Metacritic</a></li>
        <li><a href="http://www.variety.com">Variety</a></li>
    </ul>
</p>
```

RESULT

Movie Reviews:

- Empire
- Metacritic
- Variety

Linking Within Your Own Site

When you are linking to other pages within the same site, you do not need to specify the domain name in the URL. You can use a shorthand known as a ***relative*** URL. If all the pages of the site are in the same folder, then the value of the `href` attribute is just the name of the file

HTML

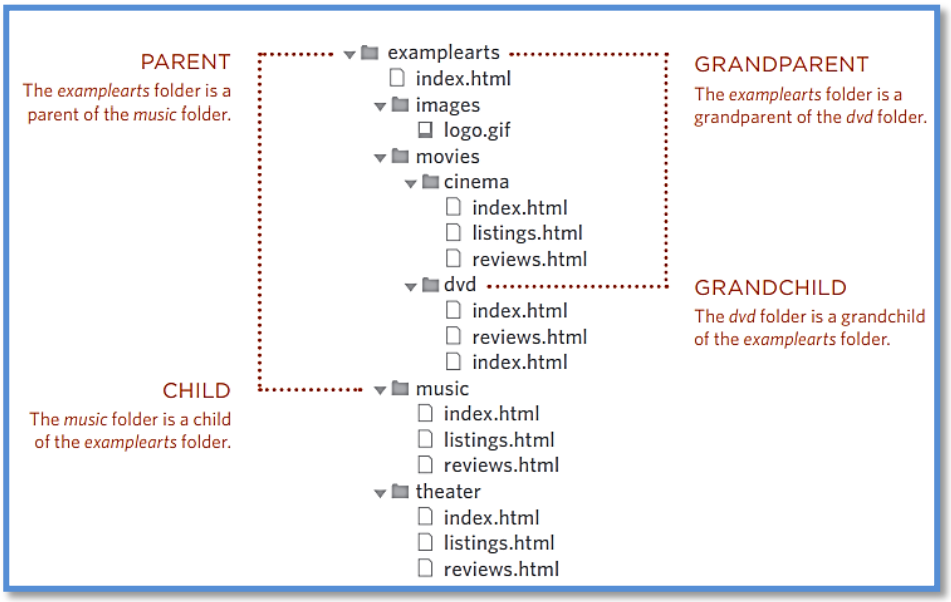
```
<p>
    <ul>
        <li><a href="index.html">Home</a></li>
        <li><a href="about-us.html">About</a></li>
        <li><a href="movies.html">Movies</a></li>
        <li><a href="contact.html">Contact</a></li>
    </ul>
</p>
```

RESULT

- Home
- About
- Movies
- Contact

Directory Structure

On larger websites it's a good idea to organize your code by placing the pages for each different section of the site into a new folder. Folders on a website are sometimes referred to as directories.



Example of a Directory Structure

Structure

The diagram shows an example of a directory structure. The top-level folder is known as the ***root*** folder. The ***root*** folder contains all of the other files and folders for a website. Each section of the site is placed in a separate folder; this helps organize the files.

Relationships

The relationship between files and folders on a website is described using the same terminology as a family tree.

The *examplearts* folder is a parent of the movies, music and theater folders. And the movies, music and theater folders are children of the *examplearts* folder

Homepages

The main homepage of a site written in HTML (and the homepages of each section in a child folder) is called `index.html`.

Web servers are usually set up to return the `index.html` file if no file name is specified. Therefore, if you enter examplearts.com it will return `examplearts.com/index.html`, and examplearts.com/music will return `examplearts .com/music/index.html`.

Relative URLs

Relative URLs can be used when linking to pages within your own website. They provide a shorthand way of telling the browser where to find your files.

This is especially helpful when creating a new website or learning about HTML because you can create links between pages when they are only on your personal computer (before you have got a domain name and uploaded them to the web).

RELATIVE LINK TYPE	EXAMPLE (from diagram on previous page)
SAME FOLDER To link to a file in the same folder, just use the file name. (Nothing else is needed.)	To link to music reviews from the music homepage `<a href="reviews.html">Reviews</a>`
CHILD FOLDER For a child folder, use the name of the child folder, followed by a forward slash, then the file name.	To link to music listings from the homepage: `<a href="music/listings.html">Listings</a>`

GRANDCHILD FOLDER
Use the name of the child folder, followed by a forward slash, then the name of the grandchild folder, followed by another forward slash, then the file name.
	To link to DVD reviews from the homepage:
	`<a href="movies/dvd/reviews.html">`
	`Reviews</a>`

PARENT FOLDER Use `../` to indicate the folder above the current one, then follow it with the file name.	To link to the homepage from the music reviews: `<a href="../index.html">Home</a>`
GRANDPARENT FOLDER Repeat the `../` to indicate that you want to go up two folders (rather than one), then follow it with the file name.	To link to the homepage from the DVD reviews: `<a href="../../index.html">Home</a>`

Email Links

To create a link that starts up the user's email program and addresses an email to a specified email address, you use the element. However, this time the value of the `href` attribute starts with `mailto:` and is followed by the email address you want the email to be sent to. When the email link is clicked on, the user's email program will open a new email message and address it to the person specified in the link.

```
<a href="mailto:jon@example.org">Email Jon</a>
```

HTML

Email Jon

RESULT

Opening Links in a New Window

If you want a link to open in a new window, you can use the `target` attribute on the opening `<a>` tag. The value of this attribute should be `_blank`.

```
<a href="http://www.imdb.com" target="_blank">
Internet Movie Database</a> (opens in new window)
```

HTML

RESULT

Internet Movie Database (opens in new window)

Linking to a Specific Part of the Same Page

Before you can link to a specific part of a page, you need to identify the points in the page that the link will go to. You do this using the id attribute (which can be used on every HTML element).

The value of the id attribute should start with a letter or an underscore (not a number or any other character) and, on a single page, no two id attributes should have the same value.

To link to an element that uses an id attribute you use the element again, but the value of the href attribute starts with the # symbol, followed by the value of the id attribute of the element you want to link to.

HTML

```
<h1 id="top">Film-Making Terms</h1>
    <a href="#arc_shot">Arc Shot</a><br />
    <a href="#interlude">Interlude</a><br />
    <a href="#prologue">Prologue</a><br /><br />
<h2 id="arc_shot">Arc Shot</h2>
<p>A shot in which the subject is photographed by an encircling camera</p>
<h2 id="interlude">Interlude</h2>
<p>A brief film scene not specifically tied to the plot, that appears within a film</p>
<h2 id="prologue">Prologue</h2>
<p>A speech, or brief scene preceding the main action or plot of a film</p>
<p><a href="#top">Top</a></p>
```

RESULT

Film-Making Terms

Arc Shot
Interlude
Prologue

Arc Shot

A shot in which the subject is photographed by an encircling camera

Interlude

A brief film scene not specifically tied to the plot, that appears within a film

Prologue

A speech, or brief scene preceding the main action or plot of a film

Top

UNIT 5 – Images

A picture can say a thousand words, and great images help make the difference between an average-looking site and a really engaging one.

Adding Images

To add an image into the page you need to use <img> element. This is an empty element (which means there is no closing tag). It must carry the following two attributes:

src	This tells the browser where it can find the image file. This will usually be a relative URL pointing to an image on your own site
alt	This provides a text description of the image which describes the image if you cannot see it.
title	You can also use the title attribute with the <img> element to provide additional information about the image. Most browsers will display the content of this attribute in a tooltip when the user hovers over the image.

HTML

```
<img src="images/quokka.jpg" alt="A family of quokka"
     title="The quokka is an Australian marsupial that is similar in size to
            the domestic cat." />
```

RESULT

Height & Width of Images

Images often take longer to load than the HTML code that makes up the rest of the page. It is, therefore, a good idea to specify the size of the image so that the browser can render the rest of the text on the page while leaving the right amount of space for the image that is still loading.

height	This specifies the height of the image in pixels.
width	This specifies the width of the image in pixels.

HTML

```
<img src="images/quokka.jpg" alt="A family of quokka"
     width="600" height="450"/>
```

UNIT 5 – Tables

A table represents information in a grid format. Examples of tables include financial reports, TV schedules, and sports results.

Grids allow us to understand complex data by referencing information on two axes.

Each block in the grid is referred to as a table cell. In HTML a table is written out row by row.

Basic Table Structure

`<table>` The `<table>` element is used to create a table. The contents of the table are written out row by row.

`<tr>` Indicate the start of each row using the opening `<tr>` tag (The tr stands for table row). It is followed by one or more elements (one for each cell in that row). At the end of the row you use a closing `</tr>` tag.

`<td>` Each cell of a table is represented using a `<td>` element. (The td stands for table data).) At the end of the row you use a closing `</td >` tag.

`<th>` The `<th>` element is used just like the `<td>` element but its purpose is to represent the heading for either a column or a row. (The th stands for table heading.) Even if a cell has no content, you should still use a `<td>` or `<th>` element to represent the presence of an empty cell otherwise the table will not render correctly.

HTML

```
<table>
        <tr>
            <td>15</td>
            <td>15</td>
            <td>30</td>
        </tr>
        <tr>
            <td>45</td>
            <td>60</td>
            <td>45</td>
        </tr>
        <tr>
            <td>60</td>
            <td>90</td>
            <td>90</td>
        </tr>
</table>
```

RESULT

15 15 30
45 60 45
60 90 90

HTML

```
<table>
    <tr>
        <th></th>
        <th scope="col">Saturday</th>
        <th scope="col">Sunday</th>
    </tr>
    <tr>
        <th scope="row">Tickets sold:</th>
            <td>120</td>
            <td>135</td>
    </tr>
    <tr>
        <th scope="row">Tickets sold:</th>
            <td>$600</td>
            <td>$675</td>
    </tr>
</table>
```

RESULT

	Saturday	Sunday
Tickets sold:	120	135
Total sales:	$600	$675

ASSESSMENT/ACTIVITIES GUIDE QUESTION/s

Guide Questions

1. What is W3C and why is it important?
2. What is the difference between a tag and an element?
3. Write out the recommended minimal structure of an HTML5 document.
4. Error Detection. Find 3 errors in the code sample

```
<html>
 <head>
      <title>Republican Candidate<title>
 </head>
 <body>
      <div align="center">
            <h1>John McCain</h1>
            <br />
            <img src="mccain.jpg" />
            <p>This photo was taken yesterday.</p>
      <div>
</html>
```

5. What is the output of the following:

```
<html>
 <head>
      <title>Links</title>
 </head>
<body>
      <p>My favorite links</p>
      <ul>
      <li><a href="http://www.nytimes.com">NYTimes</a></li>
      <li><a href="http://www.google.com/">Google</a></li>
      <li><a href="http://www.cnn.com/">CNN</a></li>
      </ul>
</body>
</html>
```

6. Create an HTML code for the following output:

HeadCol 1	HeadCol 2	HeadCol 3
Row 1, Col 1	Row 1, Col 2	Row 1, Col 3
Row 2, Col 1	Row 2, Col 2	Row 2, Col 3

<table>
<tr><td rowspan="2">Row 1-2, Col 1</td><td>Row 1, Col 2</td><td>Row 1, Col 3</td><td>Row 1, Col 4</td></tr>
<tr><td colspan="2" rowspan="2">Row 2-3, Col 2-3</td><td>Row 2, Col 4</td></tr>
<tr><td>Row 3, Col 1</td><td>Row 3, Col 4</td></tr>
<tr><td>Row 4, Col 1</td><td>Row 4, Col 2</td><td colspan="2">Row 4, Col 3-4</td></tr>
</table>

LESSON 3

HTML FORMS

INTRODUCTION

HTML borrows the concept of a form to refer to different elements that allow you to collect information from visitors to your site. Whether you are adding a simple search box to your website or you need to create more complicated insurance applications, HTML forms give you a set of elements to collect data from your users

LEARNING OUTCOMES

At the end of this module, the students are expected to:

1. How to create a form on your website
2. The different tools for collecting data
3. New HTML5 form controls

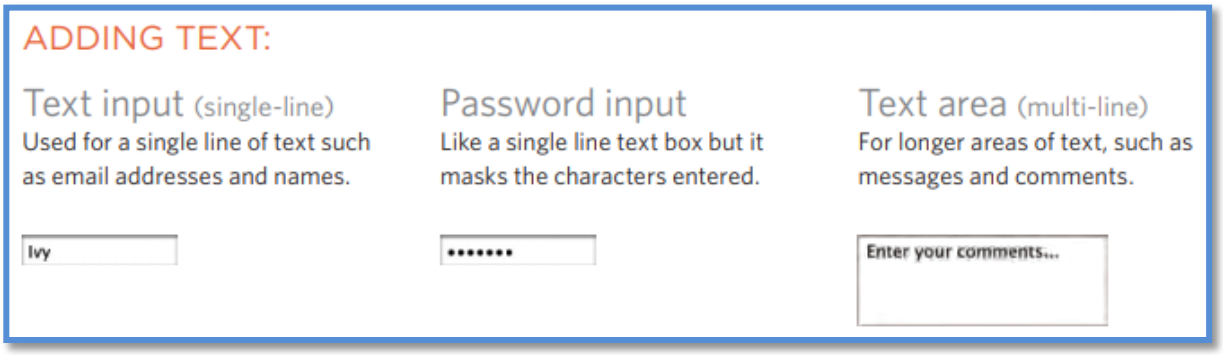
COURSE CONTENTS

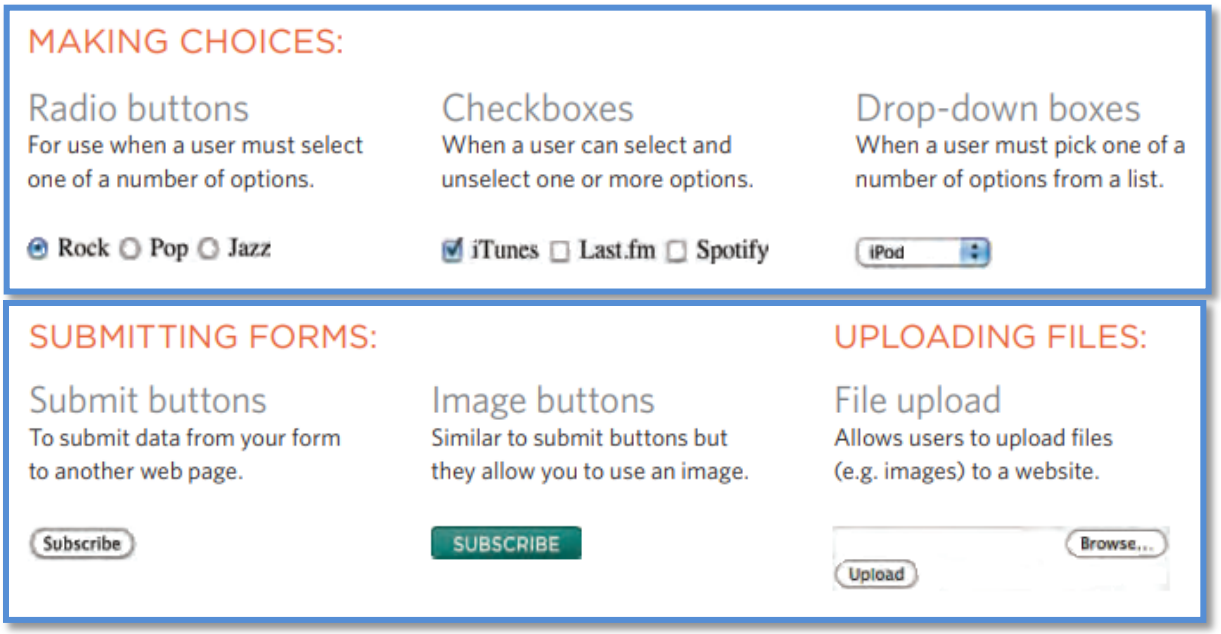
UNIT 1 – Why Forms?

Forms are used to pass data to a server. ***Forms*** are defined using the form element

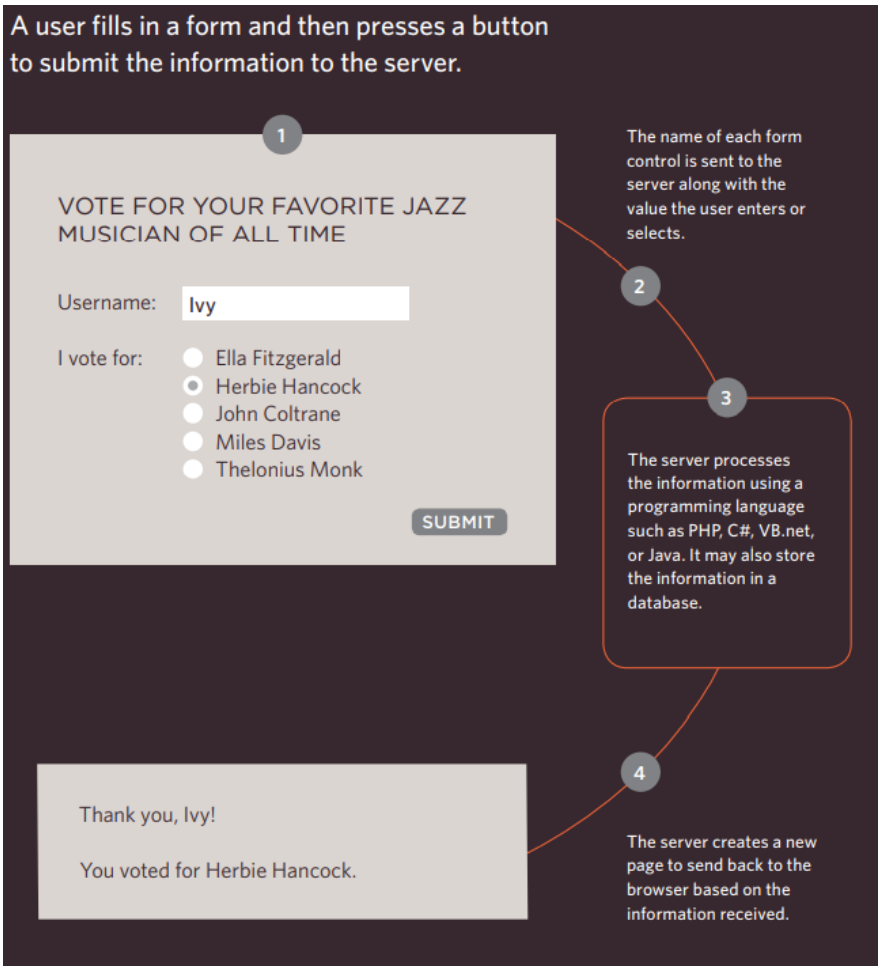
Form Controls

There are several types of form controls that you can use to collect information from visitors to your site.





How Forms Work



Form Structure

<form>	Form controls live inside a <form> element. This element should always carry the action attribute and will usually have a method and id attribute too
action	Every <form> element requires an action attribute. Its value is the name the page on the server that will receive the information in the form when it is submitted.
method	Forms can be sent using one of two methods: get or post.

HTML

```
<form action="form.html" method="get">
        <p>This is where the form controls will appear. </p>
</form>
```

RESULT

This is where the form controls will appear.

Text Input

<input>	The <input> element is used to create several different form controls. The value of the type attribute determines what kind of input they will be creating
type="text"	When the type attribute has a value of text, it creates a single line text input.
name	Each form control requires a name attribute. The value of this attribute identifies the form control and is sent along with the information they enter to the server.
maxlength	Attribute to limit the number of characters a user may enter into the text field

HTML

```
<form>
  <p>Username:
     <input type="text" name="username" size="15" maxlength="30" />
  </p>
</form>
```

RESULT

Username:

Password Input

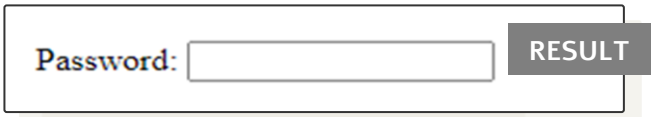
type="password" — When the type attribute has a value of password it creates a text box that acts just like a single-line text input except that character in a password field are masked (shown as asterisk or circle)

.

HTML

```
<p>Password:
   <input type="password" name="password" size="15" maxlength="30" />
</p>
```

RESULT



Text Area

<textarea> — The < textarea > element is used to create a mutli-line text input. Unlike other input elements this is not an empty element. It should therefore have an opening and a closing tag. Any text that appears between the opening <textarea> and closing </textarea> tags will appear in the text box when the page loads. If the user does not delete any text between these tags, this message will get sent to the serve

HTML

```
<form >
  <p>What did you think of this gig? </p>
    <textarea name="comments" cols="25" rows="4">Enter your
          comments...</textarea>
</form>
```

RESULT

Radio Button

type="radio" — Radio buttons allow users to pick just one of a number of options.

name — The value of the name attribute should be the same for all of the radio buttons used to answer that question

value — The value indicates the value that is sent to the server for the selected option. The value of each of the buttons in a group should be different

checked — The checked attribute can be used to indicate which value (if any) should be selected when the page loads. The value of this attribute is checked. Only one radio button in a group should use this attribute.

HTML

```
<p>Please select your favorite genre: <br />
  <input type="radio" name="genre" value="rock" checked="checked" /> Rock
  <input type="radio" name="genre" value="pop" /> Pop
  <input type="radio" name="genre" value="jazz" /> Jazz
</p>
```

RESULT

Please select your favorite genre:
◉ Rock ○ Pop ○ Jazz

Checkbox

type="radio" — Checkboxes allow users to select (and unselect) one or more options in answer to a question

HTML

```
<p>Please select your favorite spices: <br />
  <input type="checkbox" name="spice" value="salt" checked="checked"/ > salt
  <input type="checkbox" name="spice" value="pepper" />pepper
  <input type="checkbox" name="spice" value="garlic" />garlic
</p>
```

RESULT

Please select your favorite spices:
☐ salt ☐ pepper ☐ garlic

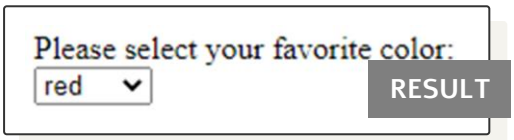
Drop Down List Box

<select> — A drop down list box (also known as a select box) allows users to select one option from a drop down list. The <select> element is used to create a drop down list box. It contains two or more <option>

<option> — The <option> element is used to specify the options that the user can select from. The words between the opening <option> and closing </option> tags will be shown to the user in the drop down box.

```
<p>Please select your favorite color: <br />
      <select name = "selColor">
          <option value = "red">red</option>
          <option value = "green">green</option>
          <option value = "blue">blue</option>
  </select>
</p>
```

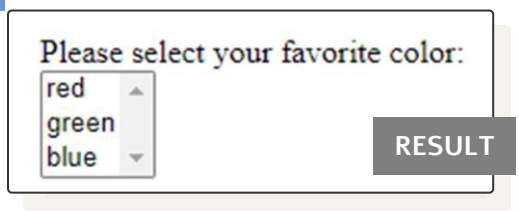
HTML



size	You can turn a drop down select box into a box that shows more than one option by adding the size attribute. Its value should be the number of options you want to show at once
multiple	You can allow users to select multiple options from this list by adding the multiple attribute with a value of multiple.

```
<p>Please select your favorite color: <br />
   <select name = "selColor" size="3" multiple >
          <option value = "red">red</option>
          <option value = "green">green</option>
          <option value = "blue">blue</option>
   </select>
</p>
```

HTML

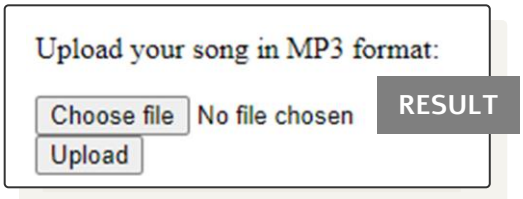


File Input Box

<input>	If you want to allow users to upload a file (for example an image, video, mp3, or a PDF), you will need to use a file input box
type="file"	This type of input creates a box that looks like a text input followed by a *browse* button. When the user clicks on the *browse* button, a window opens up that allows them to select a file from their computer to be uploaded to the website.

```
<p>Upload your song in MP3 format: </p>
<input type="file" name="song" /><br />
<input type="submit" value="Upload" />
```

HTML



Submit Button

`type="submit"` The submit button is used to send a form to the server

`value` The value attribute is used to control the text that appears on a button.

HTML

```
<p>Subscribe to our email list:</p>
<input type="text" name="email" />
<input type="submit" name="email"
       value="Subscribe" />
```

RESULT

Image Button

`type="image"` If you want to use an image for the submit button, you can give the type attribute a value of image. The `src`, `width`, `height`, and `alt` attributes work just like they do when used with the `<image>` element

HTML

```
<p>Subscribe to our email list:</p>
<input type="text" name="email" />
<input type="image"
src="images/subscribe.jpg"
 width="100" height="30" />
```

RESULT

Button & Hidden Controls

`<button>` The `<button>` element was introduced to allow users more control over how their buttons appear, and to allow other elements to appear inside the button. This means that you can combine text and images between the opening `<button>` tag and closing `</button>` tag.

`type="hidden"` These form controls are not shown on the page (although you can see them if you use the View Source option in the browser). They allow web page authors to add values to forms that users cannot see.

HTML

```
<button><img src="images/add1.gif" alt="add"
 width="10" height="10" /> Add</button>
<input type="hidden" name="bookmark"
 value="lyrics" />
```

RESULT

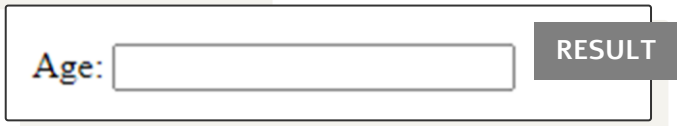
Labelling Form Controls

<label> Each form control should have its own <label> element as this makes the form accessible to vision-impaired users.

HTML

```
<label>Age: <input type="text" name="age" /></label>
```

RESULT



Grouping Form Elements

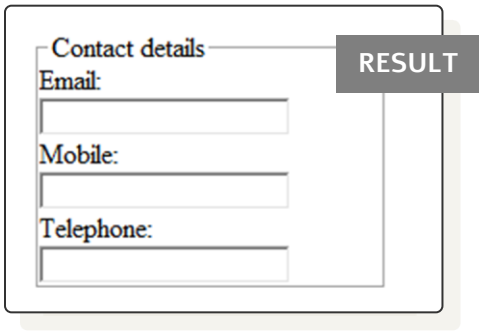
<fieldset> This element is used to group related collections of form elements into groups. It display a thin border around all of the form controls that are embedded within them

<legend> The <legend> element can come directly after opening <fieldset> tag and contains a caption which helps identify the purpose of that group of form controls.

HTML

```
<fieldset>
  <legend>Contact details</legend>
    <label>Email:<br />
    <input type="text" name="email" /></label><br />
    <label>Mobile:<br />
    <input type="text" name="mobile" /></label><br />
    <label>Telephone:<br />
    <input type="text" name="telephone" /></label>
</fieldset>
```

RESULT



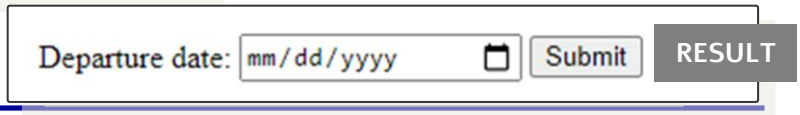
Date Input

<type="date"> If you are asking the user for a date, you can use an <input> element and give the type attribute a value of date. This will create a date input in browsers that support the new HMTL5 input types

HTML

```
<label>Departure date:<br /></label>
<input type="text" name="depart" />
<input type="submit" name="Submit" />
```

RESULT



UNIT 2 – Extra Markup

Comments in HTML

`<!-- -->` The comment tag is used to insert a comment in the HTML source code. A comment can be placed anywhere in the document and the browser will ignore everything inside the brackets. You can use comments to write notes to yourself, or write a helpful message to someone looking at your source code.

HTML

```
<p> This html comment would <!-- This
is a comment --> be displayed like this.</p>
```

RESULT

This html comment would be displayed like this.

Standard Elements Attributes

`id` an optional attribute that specifies a unique name or identifier for an element, allowing the element to be referenced elsewhere, typically by CSS or JavaScript. Each ID must be unique throughout the document

`class` an optional attribute used to define an element as being part of a class, allowing it along with all elements of that class to be referenced as a group. Any number of elements can be assigned to the same class

`style` allows you to embed inline styling inside elements

Grouping Text & Elements In a Block

`<div>` The `<div>` element allows you to group a set of elements together in one block-level box. It can also make it easier to follow your code if you have used `<div>` elements to hold each section of the page.

```
<div id="header">
<img src="images/logo.png" alt="ABC Corp" />
<ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="biography.html">Biography</a></li>
    <li><a href="works.html">Works</a></li>
    <li><a href="contact.html">Contact</a></li>
</ul>
</div><!-- end of header -->
```

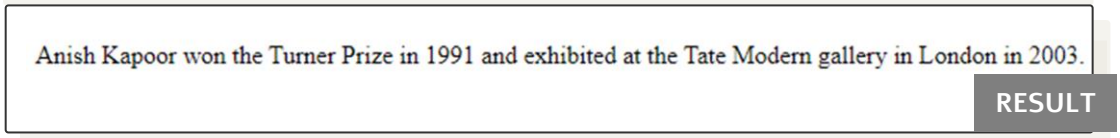
HTML

Grouping Text & Elements Inline

 The acts like an inline equivalent of the elements. It is used to either: Contain a section of text where there is no other suitable element to differentiate it from its surrounding text; Contain a number of inline elements You will usually see that a class or id attribute is used with elements to explain the purpose of the span elements and so that CSS styles can be applied to elements that have specific values for these attributes

```
<p>Anish Kapoor won the Turner Prize in 1991 and exhibited at the <span
class="gallery">Tate Modern</span> gallery in London in 2003.</p>
```

HTML



Iframes

<iframe> An iframe is like a little window that has been cut into your page — and in that window you can see another page. The term iframe is an abbreviation of inline frame. An iframe is created using the <iframe> element.

src The src attribute specifies the URL of the page to show in the frame.

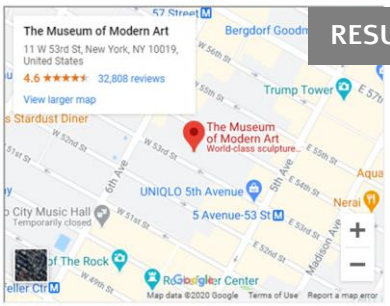
height The height attribute specifies the height of the iframe in pixels.

width The width attribute specifies the width of the iframe in pixels.

HTML

```
<iframe
width="450"
height="350"
src="http://maps.google.co.uk/maps?q=moma+new+y
ork
&amp;output=embed">
</iframe>
```

RESULT



UNIT 3 – Video and Audio

Adding a Video to a Web Page

<video>	The <video> element adds video player to a page
src	This attribute specifies the path to the video
poster	This attribute allows you to specify an image to show while the video is downloading or until the user tells the video to play
width, height	These attributes specify the size of the player in pixels.
controls	This attribute indicates that the browser should supply its own controls for playback.
autoplay	This attribute specifies that the file should play automatically
loop	Indicates that the video should start playing again once it has ended.
preload	This attribute tells the browser what to do when the page loads. It can have one of three values:

none	The browser should not load the video until the user presses play
auto	The browser should download the video when the page loads
metadata	The browser should just collect information such as the size, first frame, track list, and duration

HTML

```
<body>
    <video src="video/puppy.mp4"
        poster="images/puppy.jpg"
        width="400" height="300"
        preload
        controls
        loop>
        <p>A video of a puppy playing in the snow</p>
    </video>
</body>
```

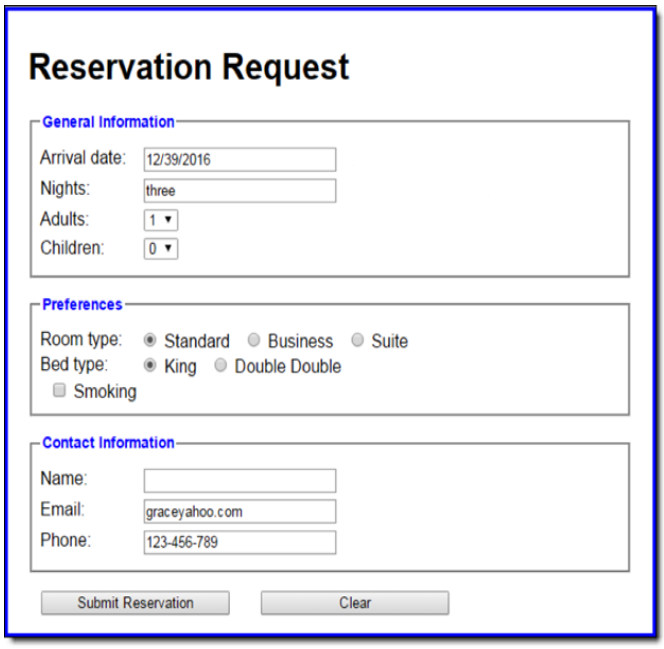
Adding Audio to Web Pages

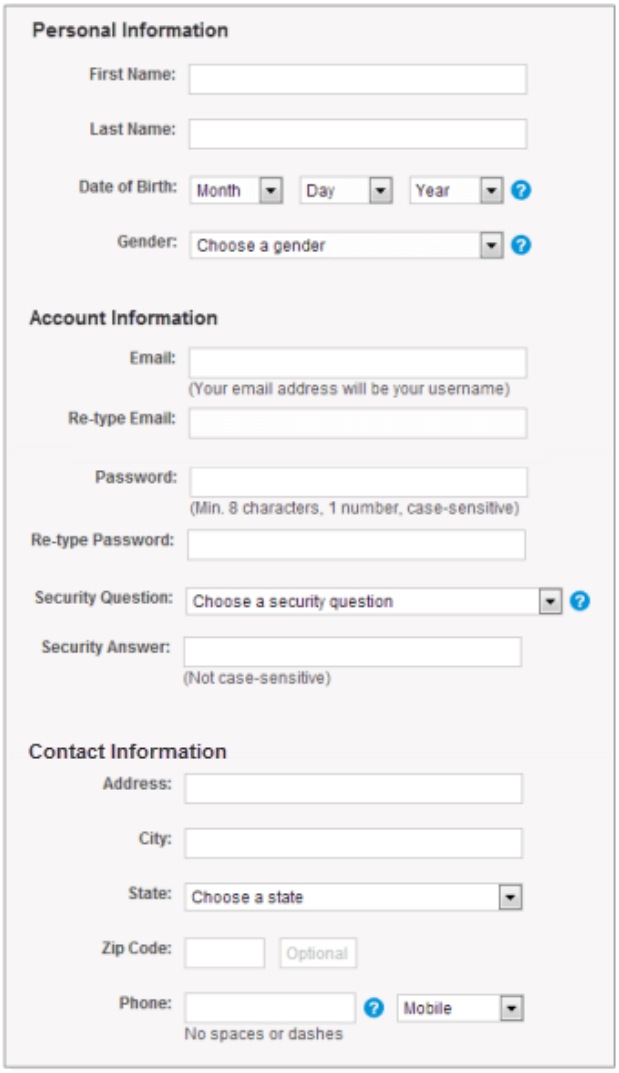
<audio>	The <video> element adds audio file to a page
src	This attribute specifies the path to the audio file
controls	This attribute indicates whether the player should display controls. If you do not use this attribute, no controls will be shown by default.
autoplay	The presence of this attribute indicates that the audio should start playing automatically
preload	This attribute indicates what the browser should do if the player is not set to autoplay.
loop	This attribute specifies that the audio track should play again once it has finished

ASSESSMENT/ACTIVITIES GUIDE QUESTION/s

Guide Questions

1. Create an HTML that will output the design on the right





2. Create an HTML that will output the design on the left.

LESSON 4

INTRODUCTION TO CSS

INTRODUCTION

In this lesson, we will look at how to make your web pages more attractive, controlling the design of them using CSS.

LEARNING OUTCOMES

At the end of this module, the students are expected to:

1. The benefits and power of Cascading Style Sheets (CSS)
2. Know how to write CSS rules
3. Know how to attach styles to the HTML document

COURSE CONTENTS

UNIT 1 – Cascading Style Sheet

Cascading Style Sheets (CSS) is the W3C standard for defining the ***presentation*** of documents written in HTML. Presentation, again, refers to the way the document is displayed or delivered to the user. With style sheets handling the presentation, HTML can handle the business of defining document structure and meaning, as intended.

The Benefits of CS

- Precise type and layout controls. You can achieve print-like precision using CSS.
- Less work. You can change the appearance of an entire site by editing one style sheet.
- More accessible sites. When all matters of presentation are handled by CSS, you can mark up your content meaningfully, making it more accessible for non-visual or mobile devices.
- Reliable browser support. Every browser in current use supports CSS Level 2 and many cool parts of CSS Level 3

How Style Sheets Work

1. Start with a document that has been marked up in HTML.
2. Write style rules for how you'd like certain elements to look.
3. Attach the style rules to the document. When the browser displays the document, it follows your rules for rendering elements

Understanding CSS: Thinking Inside the Box

The key to understanding how CSS works is to imagine that there is an invisible box around every HTML element.

Block Elements

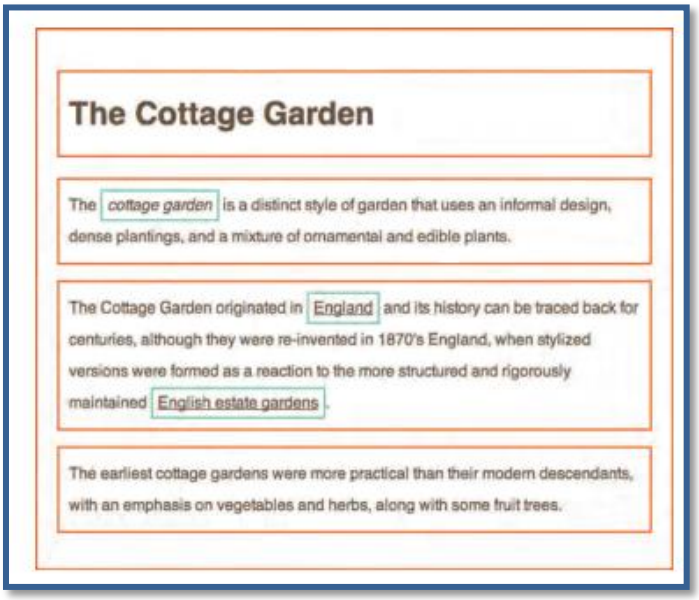
Some elements will always appear to start on a new line in the browser window. These are known as *block level* elements.

Examples of block elements are <h1>, <p>, <ul> and

Inline Elements

Some elements will always appear to continue on the same line as their neighbouring elements. These are known as *inline elements*.

Examples of block elements are <a>, <b>, and <img>



- In this example, block level elements are shown with red borders, and inline elements have green borders.
- The <body> element creates the first box, then the <h1>,<h2>,<p>,<i>, and <a> elements each create their own boxes within it.
- Using CSS, you could add a border around any of the boxes, specify its width and height, or add a background color. You could also control text inside a box — for example, its color, size, and the typeface used.

- Example Styles
  - ***Boxes***: Width and height Borders (color, width, and style) Background color and images Position in the browser window
  - ***Text***: Typeface Size Color Italics, bold, uppercase, lowercase, small-caps

Writing the Rules

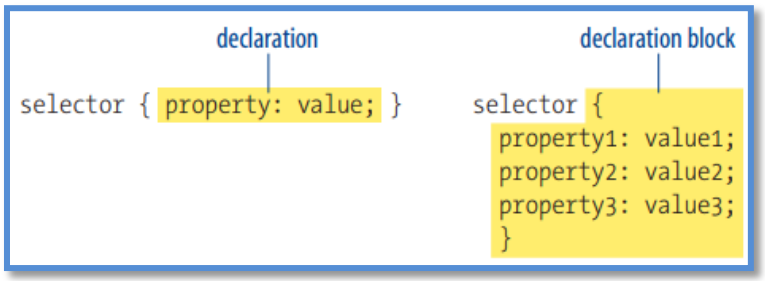
A ***style sheet*** is made up of one or more style instructions (called *rules* or *rule sets*) that describe how an element or group of elements should be displayed. Each *rule* selects an element and declares how it should look.

A CSS *rule* contains two parts: a selector and a declaration.

Selectors indicate which element the rule applies to. The same rule can apply to more than one element if you separate the element names with commas

Declarations indicate how the elements referred to in the selector should be styled. Declarations are split into two parts (a property and a value), and are separated by a colon

Syntax:



- This rule indicates that all elements should be shown in the Arial typeface.

CSS Declaration

CSS declarations sit inside curly brackets and each is made up of two parts: a property and a value, separated by a colon. You can specify several properties in one declaration, each separated by a semi-colon.

Properties indicate the aspects of the element you want to change. For example, color, font, width, height and border

Values specify the settings you want to use for the chosen properties. For example, if you want to specify a color property then the value is the color you want the text in these elements to be.

- This rule indicates that all <h1>, <h2> and <h3> elements should be shown in the Arial typeface, in a yellow color.

Using External CSS

An ***external style sheet*** is a separate, text-only document that contains a number of style rules. It must be named with the .css suffix. The .css document is then linked to or imported into one or more HTML documents. In this way, all the files in a website may share the same style sheet. This is the most powerful and preferred method for attaching style sheets to content.

<link> The < link > element can be used in an HTML document to tell the browser where to find the CSS file used to style the page. It is an empty element (meaning it does not need a closing tag), and it lives inside the <head> element. It should use three attributes:

- href This specifies the path to the CSS file (which is often placed in a folder called css or styles).
- type This attribute specifies the type of document being linked to. The value should be text/css.
- rel This specifies the relationship between the HTML page and the file it is linked to. The value should be stylesheet when linking to a CSS file

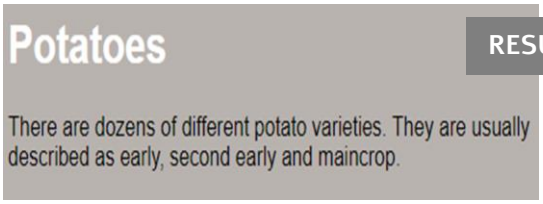
HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Using External CSS</title>
    <link href="css/styles.css" type="text/css" rel="stylesheet" />
  </head>
 <body>
     <h1>Potatoes</h1>
     <p>There are dozens of different potato varieties. They are usually described as
     early, second early and maincrop.</p>
  </body>
</html>
```

CSS

```
body {
 font-family: arial;
 background-color: rgb(185,179,175);}
h1 {
 color: rgb(255,255,255);}
```

RESULT



Using Internal CSS

You can also include CSS rules within an HTML page by placing them inside a <style> element, which usually sits inside the <head> element of the page.

The <style> element should use the type attribute to indicate that the styles are specified in CSS

HTML

```
  <head>
    <title>Using External CSS</title>
    <style type="text/css" >
            body {
                 font-family: arial;
                 background-color: rgb(185,179,175);}
            h1 {
                color: rgb(255,255,255);}
    </style>
  </head>
 <body>
     <h1>Potatoes</h1>
     <p>There are dozens of different potato varieties. They are usually described as
     early, second early and maincrop.</p>
  </body>
```

CSS Selector

In CSS, selectors are patterns used to select the element(s) you want to style.

SELECTOR	MEANING	EXAMPLE
UNIVERSAL SELECTOR	Applies to all elements in the document	`* {}` Targets all elements on the page
TYPE SELECTOR	Matches element names	`h1, h2, h3 {}` Targets the <h1>, <h2> and <h3> elements

CLASS SELECTOR	Matches an element whose `class` attribute has a value that matches the one specified after the period (or full stop) symbol	`.note {}`
		Targets any element whose `class` attribute has a value of `note`
		`p.note {}`
		Targets only <p> elements whose `class` attribute has a value of `note`

ID SELECTOR	Matches an element whose `id` attribute has a value that matches the one specified after the pound or hash symbol	`#introduction {}` Targets the element whose `id` attribute has a value of `introduction`
CHILD SELECTOR	Matches an element that is a direct child of another	`li>a {}` Targets any <a> elements that are children of an element (but not other <a> elements in the page)
DESCENDANT SELECTOR	Matches an element that is a descendent of another specified element (not just a direct child of that element)	`p a {}` Targets any <a> elements that sit inside a <p> element, even if there are other elements nested between them
ADJACENT SIBLING SELECTOR	Matches an element that is the next sibling of another	`h1+p {}` Targets the first <p> element after any <h1> element (but not other <p> elements)
GENERAL SIBLING SELECTOR	Matches an element that is a sibling of another, although it does not have to be the directly preceding element	`h1~p {}` If you had two <p> elements that are siblings of an <h1> element, this rule would apply to both

UNIT 2 – Color

Foreground Color

The foreground of an element consists of its text and border

color The color property allows you to specify the color of text inside an element. You can specify any color in CSS in one of three ways:

RGB Values These express colors in terms of how much red, green and blue are used to make it up. For example: rgb(100,100,90)

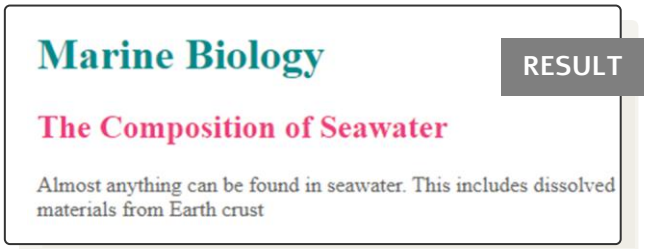
HEX Codes These are six-digit codes that represent the amount of red, green and blue in a color, preceded by a pound or hash # sign. For example: #ee3e80

Color Names There are 147 predefined color names that are recognized by browsers. For example: DarkCyan

CSS

```
/* color name */
h1 {
        color: DarkCyan;}
/* hex code */
 h2 {
        color: #ee3e80;}
 /* rgb value */
p {
        color: rgb(100,100,90);}
```

RESULT



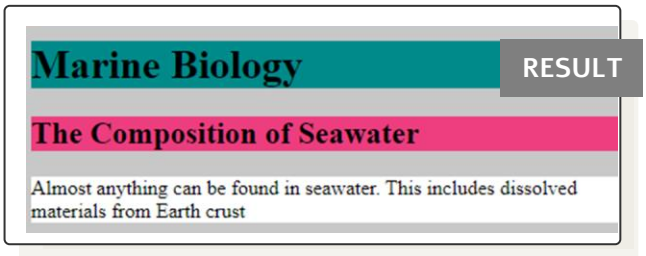
Background Color

background-color CSS treats each HTML element as if it appears in a box, and the ***background-color*** property sets the color of the background for that box.

CSS

```
body {
   background-color: rgb(200,200,200);}
h1 {
    background-color: DarkCyan;}
h2 {
    background-color: #ee3e80;}
p {
    background-color: white;}
```

RESULT



Opacity

The opacity property which allows you to specify the opacity of an element and any of its child elements. The value is a number between 0.0 and 1.0 (so a value of 0.5 is 50% opacity and 0.15 is 15% opacity).

CSS

```
body {background-color: rgb(200,200,200);}
h1 {color: green; background: white; opacity: .25;}
h2 {color: green; background: white; opacity: .5;}
h3 {color: green; background: white; opacity: 1;}
```

RESULT

Playing with Opacity

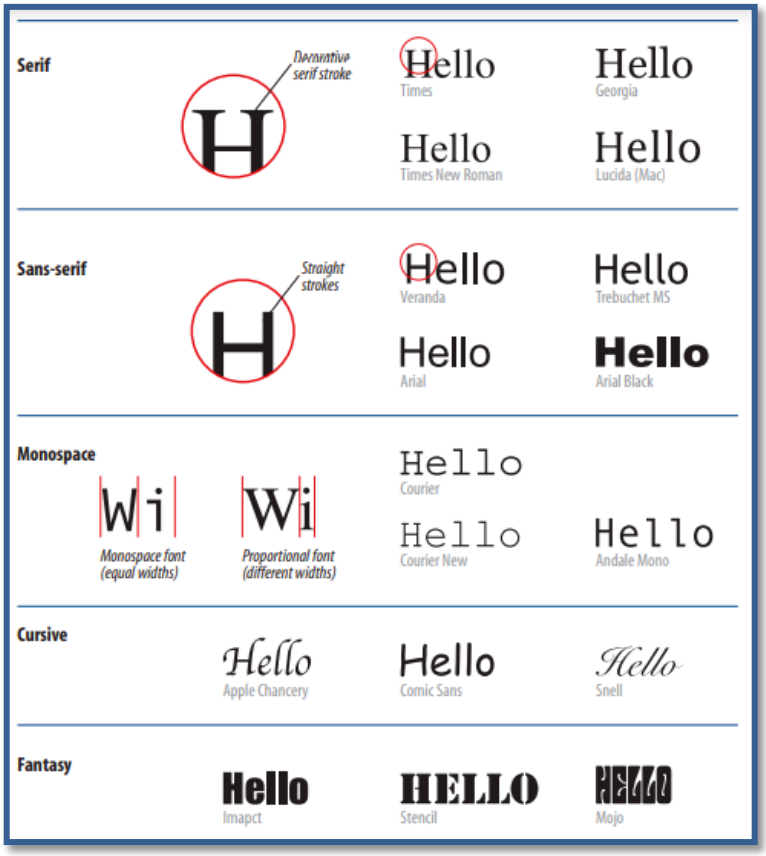
Playing with Opacity

Playing with Opacity

UNIT 3 – Text

Specifying Typefaces

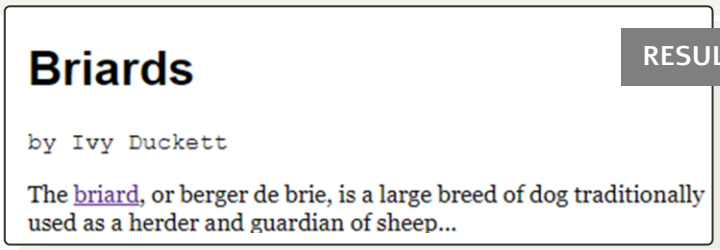
font-family The font-family property allows you to specify the typeface that should be used for any text inside the element(s) to which a CSS rule applies. The value of this property is the name of the typeface you want to use.



HTML CSS

```
<head>
   <title>Font Family</title>
   <style type="text/css" >
    body {font-family: Georgia, Times, serif;}
     h1, h2 { font-family: Arial, Verdana, sans-serif;}
     .credits { font-family: "Courier New", Courier, monospace;}
    </style>
</head>
<body>
      <h1>Briards</h1>
     <p class="credits">by Ivy Duckett</p>
      <p class="intro">The <a class="breed" href="http://en.wikipedia.org/wiki/ Briard" >
     briard</a>, or berger de brie, is a large breed of dog traditionally used as a herder and
     guardian of sheep...</p>
</body>
```

RESULT



Size of Type

font-size

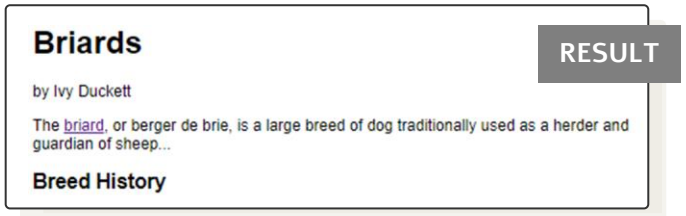
The font-size property enables you to specify a size for the font. There are several ways to specify the size of a font. The most common are:

- **Pixels**: Pixels are commonly used because they allow web designers very precise control over how much space their text takes up. The number of pixels is followed by the letters px.
- **Percentages:** The default size of text in browsers is 16px. So a size of 75% would be the equivalent of 12px, and 200% would be 32px.
- **em:** An em is equivalent to the width of a letter m.

CSS

```
body {font-family: Arial, Verdana,
        sans-serif; font-size: 12px;}
h1 {font-size: 200%;}
h2 {font-size: 1.3em;}
```

RESULT



Bold

`font-weight` The font-weight property allows you to create bold text. There are two values that this property commonly takes:

normal	This causes text to appear at a normal weight.
bold	This causes text to appear bold

```
h1 {font-weight: bold;}
```
CSS

This is an example of boldfaced RESULT

Italics

`font-style` The font-style property allows you to create italic text. There are three values that this property commonly takes:

normal	This causes text to appear at a normal style.
italic	This causes text to appear italic
oblique	This causes text to appear oblique

```
p.ital {font-style: italic;}
p.obl {font-style: oblique;}
```
CSS

This is an example of italic

This is an example of oblique RESULT

Underline & Strike

`text-decoration` The text-decoration property allows you allows you to specify the following values:

underline	This causes text to appear at a normal style.
overline	This causes text to appear italic
line-through	This causes text to appear oblique
blink	This animates the text to make it flash on and off

```
p.under{text-decoration: underline;}
p.over{text-decoration: overline;}
p.line{text-decoration: line-through;}
```
CSS

This is an example of underline

This is an example of overline

~~This is an example of line-through~~ RESULT

Alignment

`text-align` The text-align property allows you to control the alignment of text. The property can take one of four values:

`left`	This indicates that the text should be left-aligned
`right`	This indicates that the text should be right-aligned
`center`	This allows you to center text.
`justify`	This indicates that every line in a paragraph, except the last line, should be set to take up the full width of the containing box

CSS

```
h1 {text-align: left;}
p {text-align: justify;}
  .credits {text-align: right;}
```

RESULT

Responding to Users

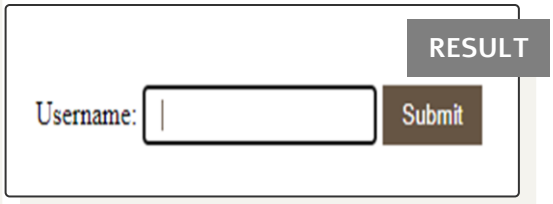
There are three pseudo-classes that allow you to change the appearance of elements when a user is interacting with them.

`:hover`	Applies when the element is selected and ready for input
`: active`	Applies when the mouse pointer is over the element
`: focus`	Applies when the element (such a link or button) is in the process of being clicked or tapped

CSS

```
input {
    padding: 6px 12px 6px 12px;
    border: 1px solid #665544;
    color: #ffffff;}
input.submit:hover {background-color: #665544;}
input.submit:active {background-color:
chocolate;}
input.text {color: #cccccc;}
input.text:focus {color: #665544;}
```

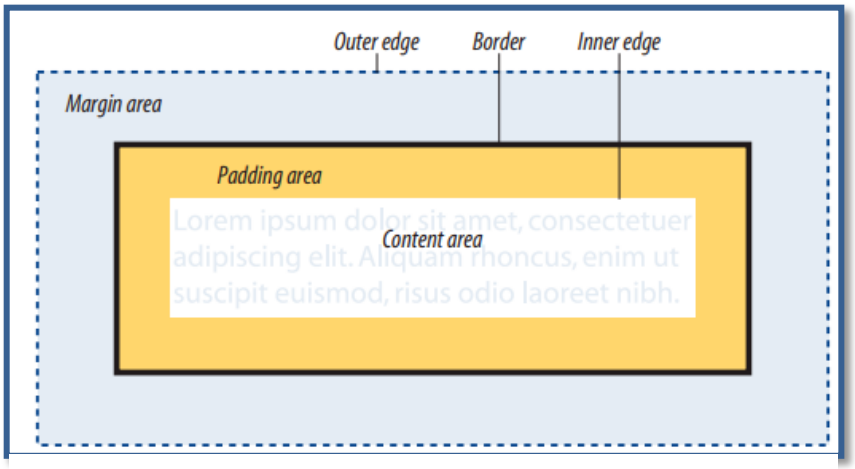
RESULT



UNIT 4 – Boxes

The Element Box

According to the box model, every element in a document generates a box to which properties such as width, height, padding, borders, and margins can be applied.



The parts of an element box according to the CSS box model

Content area	At the core of the element box is the content itself
Inner edges	The edges of the content area are referred to as the inner edges of the element box
Padding	The padding is the area held between the content area and an optional border
Border	The border is a line (or stylized line) that surrounds the element and its padding. Borders are also optional.
Margin	The margin is an optional amount of space added on the outside of the border
Outer edge	The outside edges of the margin area make up the outer edges of the element box. This is the total area the element takes up on the page, and it includes the width of the content area plus the total amount of padding, border, and margins applied to the element.

Box Dimensions

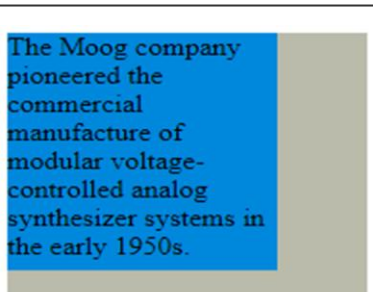
By default a box is sized just big enough to hold its contents. To set your own dimensions for a box you can use the height and width properties.

The most popular ways to specify the size of a box are to use pixels, percentages, or ems. Traditionally, pixels have been the most popular method because they allow designers to accurately control their size.

CSS

```
div.box {
   height: 200px;
   width: 200px;
   background-color: #bbbbaa;}
p {
   height: 75%;
   width: 75%;
   background-color: #0088dd;}
```

RESULT



Overflowing Content

overflow The overflow property tells the browser what to do if the content contained within a box is larger than the box itself. It can have one of two value

hidden hides any extra content that does not fit in the box.

scroll adds a scrollbar to the box so that users can scroll to see the missing content

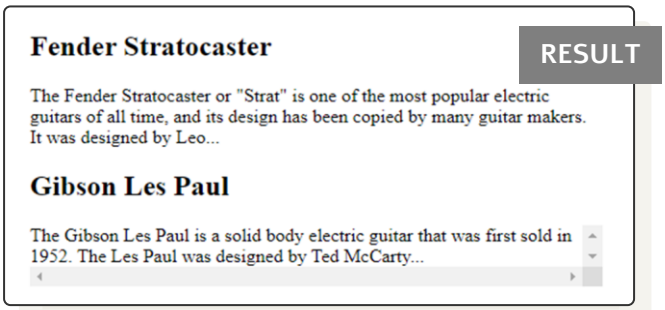
HTML

```
<h2>Fender Stratocaster</h2>
<p class="one">The Fender Stratocaster or "Strat" is one of the most popular electric guitars of all time, and its design has been copied by many guitar makers. It was designed by Leo...
</p>
<h2>Gibson Les Paul</h2>
<p class="two">The Gibson Les Paul is a solid body electric guitar that was first sold in 1952.
```

CSS

```
p.one {
    overflow: hidden;}
p.two {
    overflow: scroll;}
```

RESULT



Border

A border is simply a line drawn around the content area and its (optional) padding. You can choose from eight border styles and make them any width and color you like.

border-style

Values: none | dotted | dashed | solid | double | groove | ridge | inset | outset | inherit

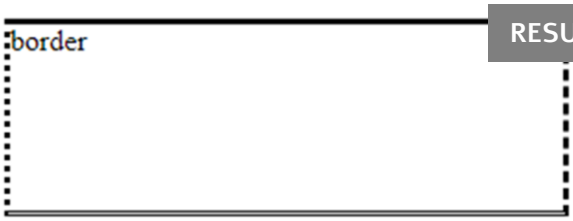
Default: none

Applies to: all elements

CSS

```
div{
    border-top-style: solid;
    border-right-style: dashed;
    border-bottom-style: double;
    border-left-style: dotted;
    width: 300px;
    height: 100px;}
```

RESULT



Margin

Margin is an optional amount of space that you can add on the outside of the border. Margins keep elements from bumping into one another or the edge of the browser window.

margin-top, margin-right, margin-bottom, margin-left

Values: length measurement | percentage | auto | inherit

Default: auto

Applies to: all elements

margin

Values: length measurement | percentage | auto | inherit

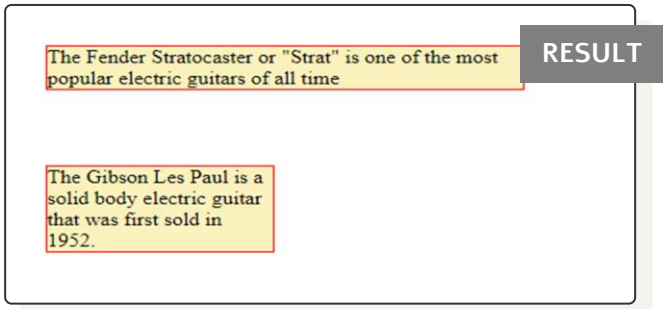
Default: auto

Applies to: all elements except elements with table display types other than table-caption, table, and inline-table

CSS

```
p.A{
    margin: 4em;
    border: 1px solid red;
    background: #FCF2BE;}
p.B{
   margin-top: 2em;
    margin-right: 250px;
    margin-bottom: 1em;
    margin-left: 4em;
    border: 1px solid red;
    background: #FCF2BE;}
```

RESULT



Padding

Padding is the space between the content area and the border

padding-top, padding-right, padding-bottom, padding-left

Values: length measurement | percentage | inherit

Default: 0

Applies to: all elements except table-row, table-row group, table-header-group, table-footer-group, table-column, and table-column-group

Padding

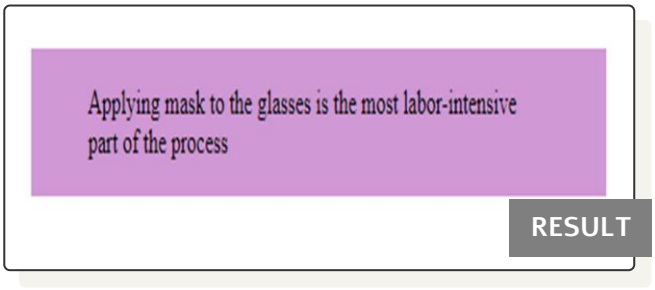
Values: length measurement | percentage | inherit

Default: 0

Applies to: all elements

```
p {
 padding-top: 1em;
 padding-right: 3em;
 padding-bottom: 1em;
 padding-left: 3em;
 background-color: #D098D4;}
```

CSS



UNIT 5 – Lists, Tables and Forms

Bullet Point Styles

`list-style-type` The list-style-type property allows you to control the shape or style of a bullet point

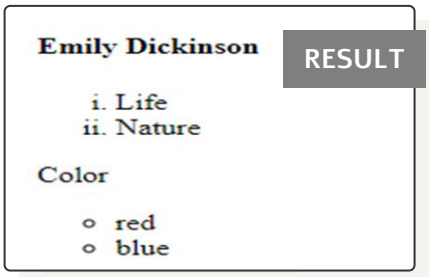
Values: none | disc | circle | square | decimal | decimal-leading-zero | lower-alpha | upper-alpha | lower-latin | upper-latin | lower-roman | upper-roman | lower-greek | inherit Default: disc

Default: disc

Applies to: ul, ol, and li (or elements whose display value is list-item)

```
ol.lr {
   list-style-type: lower-roman;}
ul.c {
   list-style-type: circle;}
```

CSS



Styling Tables

Here are some tips for styling tables to ensure they are clean and easy to follow:

Give cells padding If the text in a table cell either touches a border (or another cell), it becomes much harder to read. Adding padding helps to improve readability

Distinguish headings Putting all table headings in bold (the default style for the <th> element) makes them easier to read. You can also make headings uppercase and then either add a background color or an underline to clearly distinguish them from content.

Shade alternate rows Shading every other row can help users follow along the lines. Use a subtle distinction from the normal color of the rows to keep the table looking clean.

Align numerals You can use the text-align property to align the content of any column that contains numbers to the right, so that large numbers are clearly distinguished from smaller ones.

Border on Empty Cells

empty-cells — Use the empty-cells property to specify whether or not their borders should be shown on an empty cells. It can take one of three values:

show	This shows the borders of any empty cells.
hide	This hides the borders of any empty cells.
inherit	If you have one table nested inside another, the inherit value instructs the table cells to obey the rules of the containing table.

HTML

```
<table class="one">
  <tr>
      <td>1</td>
      <td>2</td>
  </tr>
  <tr>
      <td>3</td>
      <td></td>
  </tr>
</table>
```

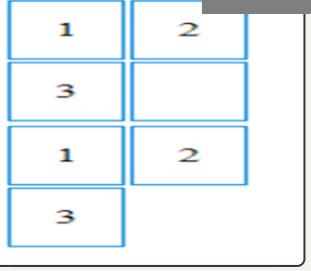
HTML

```
<table class="two">
  <tr>
      <td>1</td>
      <td>2</td>
  </tr>
  <tr>
      <td>3</td>
      <td></td>
  </tr>
</table>
```

CSS

```
td {
    border: 1px solid
#0088dd;
    padding: 15px;}
table.one {
    empty-cells: show;}
table.two {
    empty-cells: hide;}
```

RESULT



Gaps Between Cells

border-spacing — The border-spacing property allows you to control the distance between adjacent cells. By default, browsers often leave a small gap between each table cell, so if you want to increase or decrease this space then the border-spacing property allows you to control the gap.

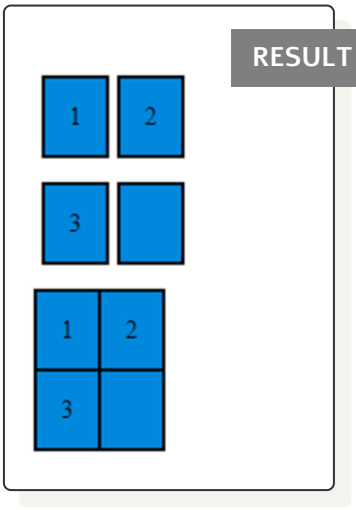
border-collapse — When a border has been used on table cells, where two cells meet, the width of lines would be twice that of the outside edges. It is possible to collapse adjacent borders to prevent this using the border-collapse property. Possible values are:

collapse	Borders are collapsed into a single border where possible
separate	Borders are detached from each other

CSS

```
td {
   background-color:
   #0088dd;
   padding: 15px;
   border: 2px solid
   #000000;}
table.one {
   border-spacing: 5px
   15px;}
table.two {
   border-collapse:
   collapse;}
```

RESULT



Styling Text Inputs

font-size — sets the size of the text entered by the user.

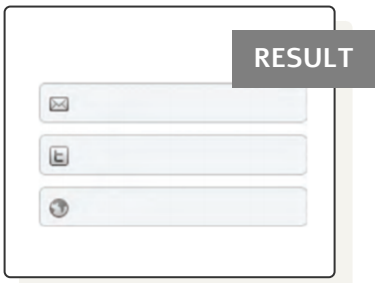
color — sets the text color, and background-color sets the background color of the input.

border — adds a border around the edge of the input box, and border-radius can be used to create rounded corners (for browsers that support this property)

CSS

```
input {
   font-size: 120%; color: #5a5854;
   background-color: #f2f2f2;
   border: 1px solid #bdbdbd;
   border-radius: 5px;
   padding: 5px 5px 5px 30px;
   background-repeat: no-repeat;
   background-position: 8px 9px;
   display: block; margin-bottom: 10px;}
input:focus {
   background-color: #ffffff; border: 1px solid#b1e1e4;}
input#email {
   background-image: url("images/email.png");}
input#twitter {
   background-image: url("images/twitter.png");}
input#web { background-image: url("images/web.png ");}
```

RESULT



Styling Submit Buttons

color — is used to change the color of the text on the button.

text-shadow	can give a 3D look to the text in browsers that support this property.
border-bottom	has been used to make the bottom border of the button slightly thicker, which gives it a more 3D feel.
background-color	can make the submit button stand out from other items around it.

Styling Fieldset and Legend

width	is used to control the width of the fieldset. In this example, the width of the fieldset forces the form elements to wrap onto a new line in the correct place.
color	is used to control the color of text.
background-color	is used to change the color behind these items.
border	is used to control the appearance of the border around the fieldset and/or legend.
border-radius	is used to soften the edges of these elements in browsers that support this property.
padding	can be used to add space inside these elements

UNIT 6 – Layout

Controlling the Position of Elements

CSS has the following positioning schemes that allow you to control the layout of a page: ***normal flow***, relative positioning, and ***absolute positioning***. You specify the positioning scheme using the position property in CSS. You can also float elements using the float property.

Normal Flow

Every block-level element appears on a new line, causing each item to appear lower down the page than the previous one. Even if you specify the width of the boxes and there is space for two elements to sit side-byside, they will not appear next to each other. This is the default behavior

HTML

```
<body>
  <h1>The Evolution of the Bicycle</h1>
  <p>In 1817 Baron von Drais invented a walking
   machine that would help him get around the
   royal gardens faster...</p>
</body>
```


```
body {
   width: 750px;
   font-family: Arial,
   Verdana, sans-serif;
   color: #665544;}
h1 {
 background-color:
#efefef;
 padding: 10px;}
p {
    width: 450px;}
```

CSS

Relative Positioning

This moves an element from the position it would be in normal flow, shifting it to the top, right, bottom, or left of where it would have been placed. This does not affect the position of surrounding elements; they stay in the position they would be in in normal flow.

position:relative — You can indicate that an element should be relatively positioned using the position property with a value of relative

```
p.example{
    position: relative;
    top: 10px;
    left: 100px;}
```

CSS

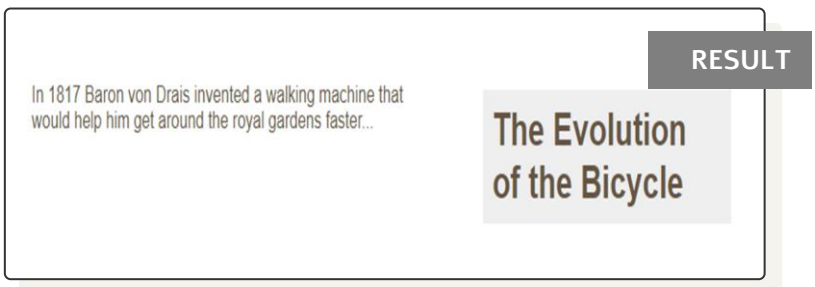
Absolute positioning

This positions the element in relation to its containing element. It is taken out of normal flow, meaning that it does not affect the position of any surrounding elements

position:absolute — the box is taken out of normal flow and no longer affects the position of other elements on the page.

```
h1 {
   position: absolute; top:
   0px; left: 500px; width:
   250px;}
 p {
   width: 450px;}
```

CSS



Fixed Positioning

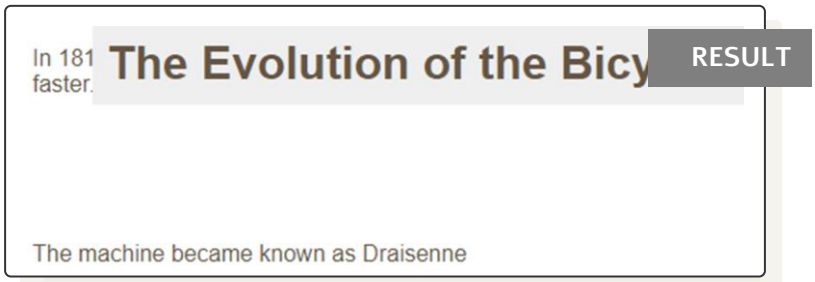
This is a form of absolute positioning that positions the element in relation to the browser window, as opposed to the containing element. Elements with fixed positioning do not affect the position of surrounding elements and they do not move when the user scrolls up or down the page

position:fixed — It positions the element in relation to the browser window. Therefore, when a user scrolls down the page, it stays in the exact same place

CSS

```
h1 {
   position: fixed; top: 0px;
   left: 50px; padding: 10px;
   margin: 0px; width: 100%;
   background-color:
   #efefef;}
p.example{
   margin-top: 100px;}
```

RESULT



Floating Elements

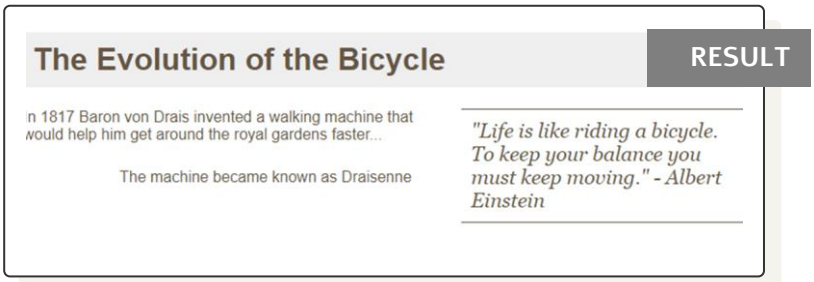
Floating an element allows you to take that element out of normal flow and position it to the far left or right of a containing box. The floated element becomes a block-level element around which other content can flow

float — The float property allows you to take an element in normal flow and place it as far to the left or right of the containing element as possible.

CSS

```
blockquote {
    float: right;
    width: 275px;
    font-size: 130%;
    font-style: italic;
    font-family: Georgia,
   Times, serif;
    margin: 0px 0px 10px
10px;
    padding: 10px;
    border-top: 1px solid
#665544;
    border-bottom: 1px
solid #665544;}
```

RESULT



Creating Multi-Column Layouts with Floats

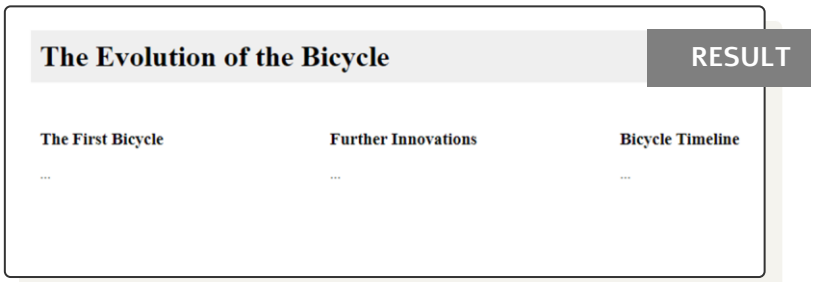
Many web pages use multiple columns in their design. This is achieved by using a element to represent each column.

width	This sets the width of the columns
float	This positions the columns next to each other.
margin	This creates a gap between the columns.

CSS

```
.column1of2 {
    float: left;
     width: 620px;
     margin: 10px;}
.column2of2 {
    float: left;
    width: 300px;
    margin: 10px;}
```

RESULT



UNIT 7 – Images

Size of Images

You can control the size of an image using the width and height properties in CSS, just like you can for any other box

HTML

```
<img src="images/magnolia-large.jpg"
 class="large" alt="Magnolia" />
<img src="images/magnolia-medium.jpg"
 class="medium" alt="Magnolia" />
<img src="images/magnolia-small.jpg"
 class="small" alt="Magnolia" />
```

CSS

```
img.large {
   width: 500px;height: 500px;}
img.medium {
   width: 250px;height: 250px;}
img.small {
   width: 100px;height: 100px;}
```

RESULT

Aligning Images Using CSS

Web page authors are increasingly using the float property to align images. There are two ways that this is commonly achieved:

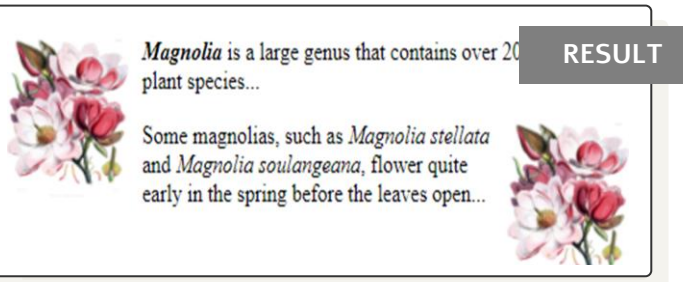
1: The float property is added to the class that was created to represent the size of the image

2: New classes are created with names such as align-left or align-right to align the images to the left or right of the page. These class names are used in addition to classes that indicate the size of the image

CSS

```
img.align-left {
    float: left;margin-right: 10px;}
img.align-right {
   float: right;margin-left: 10px;}
img.medium {
   width: 100px; height: 100px;}
```

RESULT



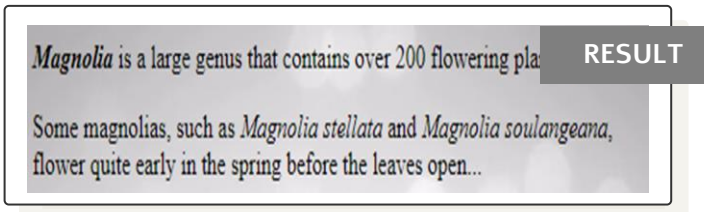
Background Images

background-image — The background-image property allows you to place an image behind any HTML element. This could be the entire page or just part of the page. By default, a background image will repeat to fill the entire box.

CSS

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

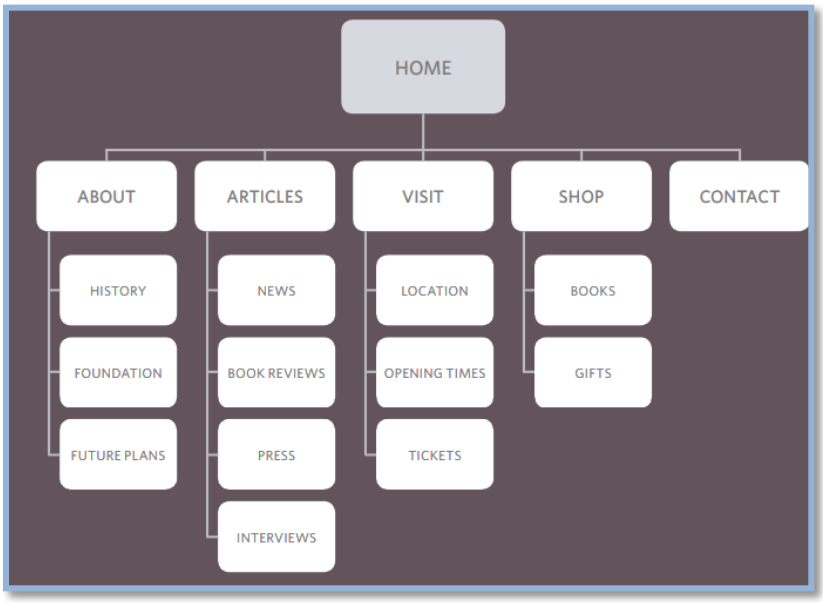
RESULT



UNIT 8 – Process and Design

Site Maps

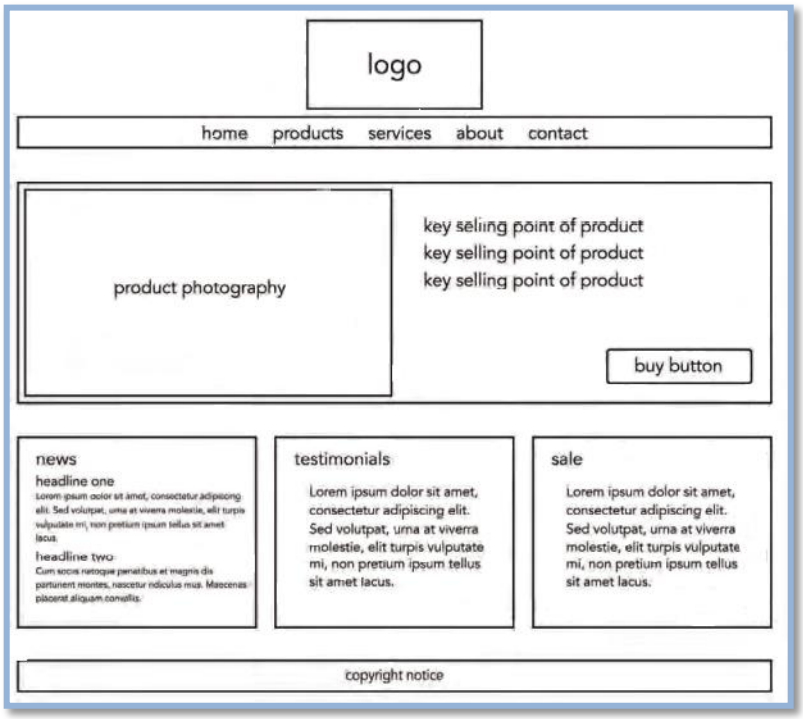
The aim is to create a diagram of the pages that will be used to structure the site. This is known as a site map. A site map will usually begin with the homepage. Additionally, if the site is large and is compartmentalized into sections, each section might require its own section homepage to link to all of the information within it.



Example of a Site Map

Wireframes

A wireframe is a simple sketch of the key information that needs to go on each page of a site. It shows the hierarchy of the information and how much space it might require.



Example of a Wireframe

Visual Hierarchy

Visual hierarchy refers to the order in which your eyes perceive what they see. It is created by adding visual contrast between the items being displayed. Items with higher contrast are recognized and processed first.

Size Larger elements will grab users' attention first. For this reason it is a good idea to make headings and key points relatively large

Color Foreground and background color can draw attention to key messages. Brighter sections tend to draw users' attention first.

Style An element may be the same size and color as surrounding content but have a different style applied to it to make it stand out

Images Images create a high visual contrast and often attract the eye first. They can be used to draw attention to a specific message within the page

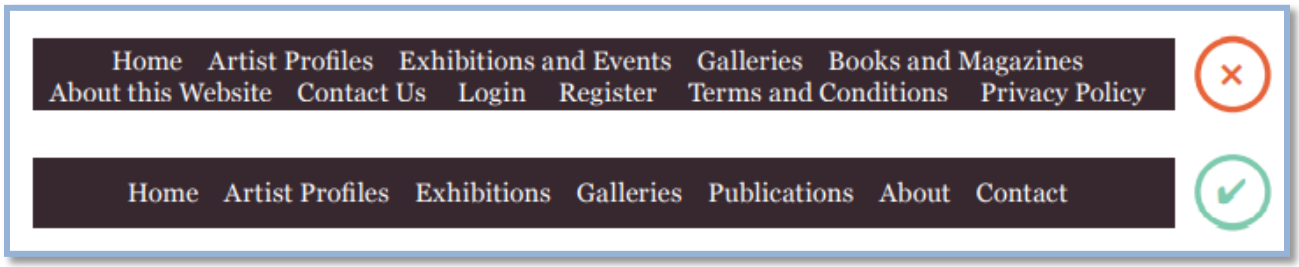
Designing Navigation

Site navigation not only helps people find where they want to go, but also helps them understand what your site is about and how it is organized. Good navigation tends to follow these principles.

Concise Ideally, the navigation should be quick and easy to read. It is a good idea to try to limit the number of options in a menu to no more than eight links. These can link to section homepages which in turn link to other pages

Clear Users should be able to predict the kind of information that they will find on the page before clicking on the link.

Selective The primary navigation should only reflect the sections or content of the site. Functions like logins and search, and legal information like terms and conditions and so on are best placed elsewhere on the page



Context Good navigation provides context. It lets the user know where they are in the website at that moment. Using a different color or some kind of visual marker to indicate the current page is a good way to do this

Interactive Each link should be big enough to click on and the appearance of the link should change when the user hovers over each item or clicks on it. It should also be visually distinct from other content on the page

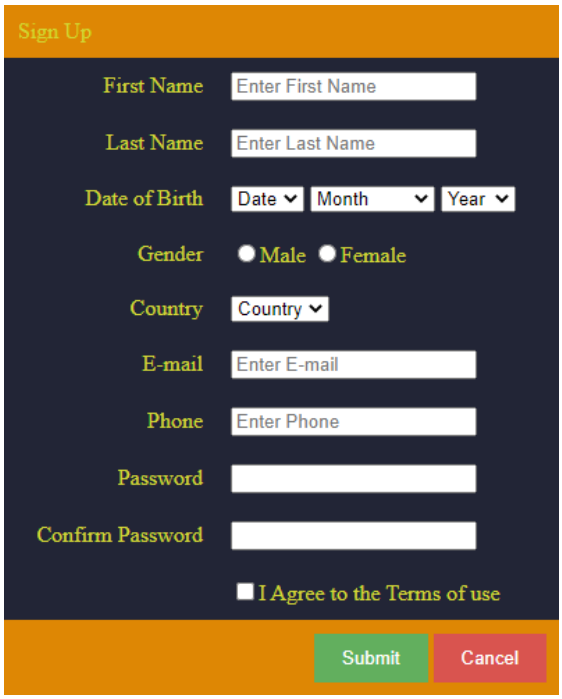
Consistent The more pages a site contains, the larger the number of navigation items there will be. Although secondary navigation will change from page to page, it is best to keep the primary navigation exactly the same.

ASSESSMENT/ACTIVITIES GUIDE QUESTION/s

Guide Questions

1. What are the limitations of CSS ?
2. What are the advantages of CSS ?
3. What are the ways can a CSS be integrated as a web page?
4. Discuss the advantages and disadvantages of Embedded Style Sheet
5. What does CSS selector mean?
6. Differentiate Style Sheet concept from HTML?
7. What are the various fonts' attributes and what does it represents?
8. What is CSS Box Model and what are its elements?
9. Compare RGB values with Hexadecimal color codes ?
10. What are the various ways of using CSS in an HTML page?

11. Create and HTML and CSS that will output the design below:



LESSON 5

INTRODUCTION TO JAVASCRIPT

INTRODUCTION

JavaScript is a very powerful client-side scripting language. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage more lively and interactive, with the help of JavaScript..

LEARNING OUTCOMES

At the end of this module, the students are expected to:

1. Understand the Javascript
2. Know how to declare variable and arrays in javascript
3. Learn the if/else statements and loops

COURSE CONTENTS

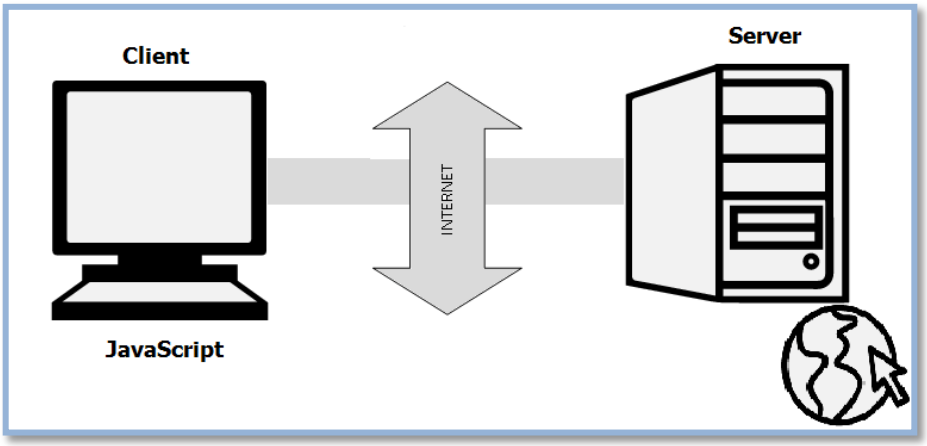
UNIT 1 – JavaScript

What is Javascript?

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

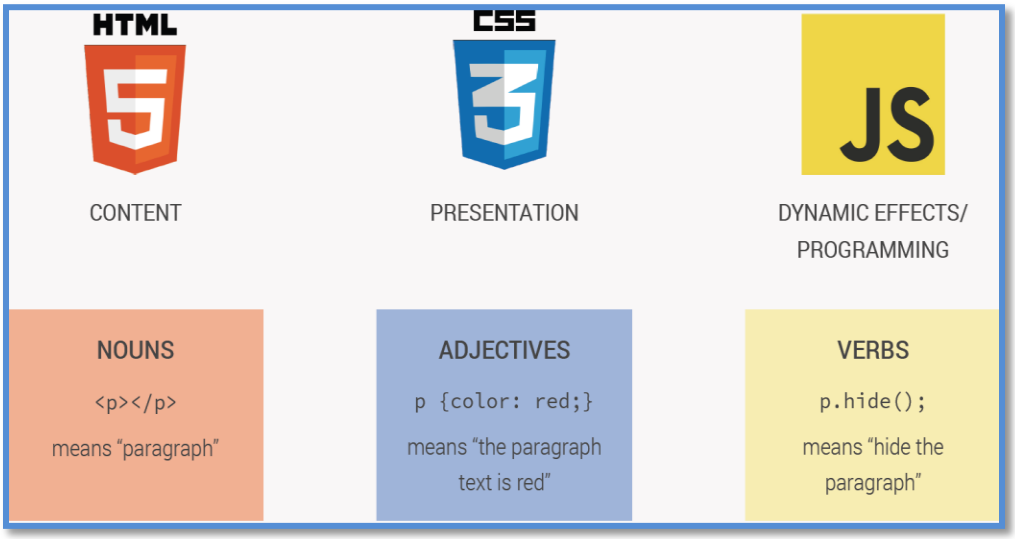
JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform



Advantages of JavaScript

- **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.



Limitations of JavaScript

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multi-threading or multiprocessor capabilities.

Java and JavaScript

Similarities

- both are interpreted, not compiled
- both are relaxed about syntax, rules, and types
- both are case-sensitive
- both have built-in regular expressions for powerful text processing

Difference

- JavaScript has a different object model from Java
- JavaScript is not strongly typed

Working with JavaScript

JavaScripts are inserted into HTML pages using the script element. You can place the `<script>` tags containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the `<head>` tags.

Syntax:

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

`type` This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

HTML

```
<body>
  <script type = "text/javascript">
        document.write("Hello World!")
  </script>
</body>
```

RESULT

Hello World!

Adding JavaScript to a Page

You can embed a script right in a document or keep it in an external file and link it to the page. Both methods use the script element.

Embedded script

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

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

External scripts

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

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

UNIT 2 – Javascript Basic

Whitespace and Line Breaks

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

Semicolons are Optional

Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line

```
<script type="text/javascript">
        var1 =10
        var2 =10
</script>
```

Can be written without semicolon

```
<script type="text/javascript">
        var1 =10; var2 =10
</script>
```

When formatted in a single line as follows, you must use semicolons

NOTE: It is a good practice to use semicolon

Case Sensitivity

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

NOTE: Care should be taken while writing variable and function names in Javascript

Comments in JavaScript

JavaScript supports both C-style and C++-style comments, Thus –

- Any text between a `//` and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters `/*` and `*/` is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence `<!--`. JavaScript treats this as a single-line comment, just as it does the `//` comment.
- The HTML comment closing sequence `-->` is not recognized by JavaScript so it should be written as `//-->`.

```
<script type="text/javascript">
   // This is a comment. It is similar to comments in C++
   /*
    * This is a multi-line comment in JavaScript
    * It is very similar to comments in C Programming
    */
</script>
```

Data Types

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

JavaScript allows you to work with three primitive data types –

- **Numbers:** numeric values. *eg.* 123, 120.50 etc.
- **Strings**: A string of text enclosed in matching quotation marks. *e.g.* "This text string" etc.
- **Boolean** e.g. true or false.

JavaScript also defines two trivial data types

- **Null:** A single value, null
- **undefined:** If we declare a variable by giving it a name but no value, that variable contains a value of "undefined."

Variables

Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container. There are a few rules around variable naming:

- It must start with a letter or an underscore.
- It may contain letters, digits, and underscores in any combination.
- It may not contain character spaces. As an alternative, use underscores in place of spaces or close up the space and use camel case instead (for example, my_variable or myVariable).
- It may not contain special characters (! . , / \ + * = etc.)

Declaring Variables

- JavaScript is dynamically typed, that is, variables do not have declared types
- JavaScript has variables that you can declare with the optional var keyword
- Variables declared within a function are local to that function
- Variables declared outside of any function are global variables

```
<script type="text/javascript">
        var amount;
        var name;
</script>
```

```
<script type="text/javascript">
        var amount, name;
</script>
```

Variable Initialization

Storing a value in a variable is called ***variable initialization***. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create a variable named money and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

```
<script type="text/javascript">
   var name = "Ali";
   var amount;
   amount = 2000.50;
</script>
```

NOTE: Use the var keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

Variable Typing

JavaScript is a very loosely typed language. Variables do not have declared types.The *type* of a variable is determined only when a value is assigned and can change as the variable appears in different contexts

```
<script type= "text/javascript">
    n = '838102050' // Set 'n' to a string
    document.write('n = ' + n + ', and is a ' + typeof n + '<br />')
    n = 12345 * 67890; // Set 'n' to a number
    document.write('n = ' + n + ', and is a ' + typeof n + '<br />')
    n += ' plus some text' // Change 'n' from a number to a string
    document.write('n = ' + n + ', and is a ' + typeof n + '<br />')
</script>
```

RESULT

```
n = 838102050, and is a string
n = 838102050, and is a number
n = 838102050 plus some text, and is a string
```

JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

Global Variables − A global variable has global scope which means it can be defined anywhere in your JavaScript code.

Local Variables − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable with the same name

```
<script type= "text/javascript">
      var myVar = "global";      // Declare a global variable
        function checkscope( ) {
          var myVar = "local";    // Declare a local variable
          document.write(myVar);}
</script>
```

Expressions

Expression is a combination of values, variables, operators, and functions that results in a value; the result can be a number, a string, or a Boolean value

```
<script type= "text/javascript">
   document.write('a = ' + (42 > 3)  + '<br />')
   document.write('b = ' + (91 < 4) + '<br />')
   document.write('c = ' + (8 == 2) + '<br />')
   document.write('d = ' + (4 < 17) + '<br />')
</script>
```

JavaScript Operators

Operator	Description
() [] .	Parentheses, call, and member
++ --	Increment/decrement
+ - ~ !	Unary, bitwise, and logical
* / %	Arithmetic
+ -	Arithmetic and string
<< >> >>>	Bitwise
< > <= >=	Comparison
== != === !==	Comparison
&&	Logical
||	Logical
? :	Ternary
= += -= *= /= %=	Assignment
<<= >>= >>>= &= ^= |= ,	Sequential evaluation

UNIT 3 – Conditional Statement

The if Statement

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax:

```
if (expression) {
        Statement(s) to be executed if expression is true
}
```

```
<body>
  <script type = "text/javascript">
     var age = 20;
      if( age > 18 ) {
         document.write("<b>Qualifies for driving</b>");}
   </script>
   <p>Set the variable to different value and then try...</p>
</body>
```

RESULT

Qualifies for driving

Set the variable to different value and then try...

The if...else statement

The if…else statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Syntax:

```
if (expression) {
        Statement(s) to be executed if expression is true
} else {
        Statement(s) to be executed if expression is true
}
```

```
<script type = "text/javascript">
   var age = 15;
    if( age > 18 ) {
       document.write("<b>Qualifies for driving</b>");
     } else {
       document.write("<b>Does not qualifies for driving</b>");
     }
</script>
<p>Set the variable to different value and then try...</p>
```

RESULT

Does not qualify for driving

Set the variable to different value and then try...

The Switch Case Statement

The objective of a switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

Syntax:

```
switch (expression) {
  case condition 1: statement(s)
  break;
  case condition 2: statement(s)
  break;
  case condition n: statement(s)
  break;
  default: statement(s)
}
```

```
<script type = "text/javascript">
        var grade = 'A';
        document.write("Entering switch block<br />");
        switch (grade)
        {
          case 'A': document.write("Good job<br />");
          break;
          case 'B': document.write("Pretty good<br />");
          break;
          case 'C': document.write("Passed<br />");
          break;
          default:  document.write("Unknown grade<br />")
        }
 </script>
```

RESULT

```
Entering switch block
Good job
Exiting switch block

Set the variable to different value and then try...
```

UNIT 4 – Looping

The while Loop

The most basic loop in JavaScript is the while loop which would be discussed in this chapter. The purpose of a while loop is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

Syntax:

```
while (expression) {
        Statement(s) to be executed if expression is true
}
```

```
<script type = "text/javascript">
   var count = 0;
   document.write("Starting Loop ");
   while (count < 5){
      document.write("Current Count : " + count + "<br />");
      count++;
    }
    document.write("Loop stopped!");
 </script>
 <p>Set the variable to different value and then try...</p>
```

RESULT

```
Starting Loop Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Loop stopped!

Set the variable to different value and then try...
```

The do...while Loop

The do...while loop is similar to the while loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is false.

Syntax:

```
do {
        Statement(s) to be executed
} while (expression);
```

```
<script type = "text/javascript">
   var count = 0;
   document.write("Starting Loop ");
   do {
      document.write("Current Count : " + count + "<br />");
      count++;
    }   while (count < 5)
    document.write("Loop stopped!");
 </script>
 <p>Set the variable to different value and then try...</p>
```

RESULT

```
Starting Loop Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Loop stopped!

Set the variable to different value and then try...
```

The for Loop

The for loop is the most compact form of looping. It includes the following three important parts

- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

Syntax:

```
for (initialization; test condition; iteration statement){
        Statement(s) to be executed if expression is true
}
```

```
<script type = "text/javascript">
   var count;
   document.write("Starting Loop ");
   for (count=0; count<5; count++) {
      document.write("Current Count : " + count + "<br />");
      count++;
    }
    document.write("Loop stopped!");
 </script>
 <p>Set the variable to different value and then try...</p>
```

RESULT

```
Starting Loop Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Loop stopped!

Set the variable to different value and then try...
```

The Loop Control

JavaScript provides full control to handle loops and switch statements. There may be a situation when you need to come out of a loop without reaching its bottom. There may also be a situation when you want to skip a part of your code block and start the next iteration of the loop

The break Statement

The break statement, which was briefly introduced with the switch statement, is used to exit a loop early, breaking out of the enclosing curly braces.

The continue Statement

The continue statement tells the interpreter to immediately start the next iteration of the loop and skip the remaining code block. When a continue statement is encountered, the program flow moves to the loop check expression immediately and if the condition remains true, then it starts the next iteration, otherwise the control comes out of the loop.

UNIT 5 – Javascript Functions

Function

A ***function*** is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. ***Functions*** allow a programmer to divide a big program into a number of small and manageable functions.

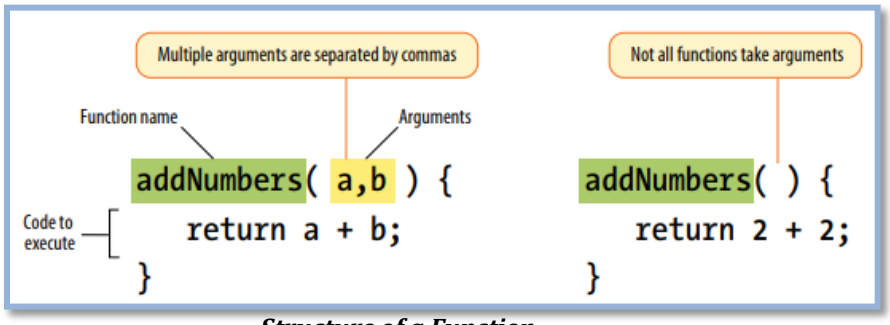
Defining a Function

The most common way to define a function in JavaScript is by using the ***function*** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax:

```
function functionname (parameter-list) {
      statements
    }
```

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



Structure of a Function

Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function

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

RESULT

Click the following button to call the function

Say Hello

RESULT

Hello there!

Function Parameters

There is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

```
<head>
    <script type = "text/javascript">
      function sayHello(name,age) {
        document.write (name + " is " + age + " years old.");
      }
    </script>
 </head>
 <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello('Ana',18)" value = "Say Hello">
    </form>
  </body>
```

RESULT

Click the following button to call the function

Say Hello

RESULT

Ana is 18 years old.

The return Statement

The return statement is used to specify the value that is returned from the function. Functions that are going to return a value must use the return statement. This statement should be the last statement in a function.

```
<script type = "text/javascript">
   function product(a,b) {
        return a*b;
     }
   document.write("The product of 4 and 3 is " + product(4, 3))
</script>
```

RESULT

The product of 4 and 3 is 12

UNIT 6 – Javascript Events

What is an Event ?

Events are things that occur within the browser. Web browsers know how to recognize events and respond to them.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc

Every HTML element contains a set of ***events*** which can trigger JavaScript Code.

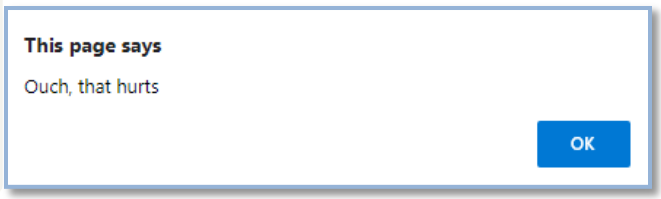
Different Types of Javascript Events

Event	Handler	This event occurs when:
abort	onabort	An action is aborted
blur	onblur	An item loses focus
change	onchange	A control's data is changed
click	onclick	When an element is clicked
dblclick	ondblclick	When an element is double clicked
dragdrop	ondragdrop	An element is dragged and dropped
error	onerror	A JavaScript error occurs
focus	onfocus	An element receives focus
keydown	onkeydown	A keyboard key is held down
keypress	onkeypress	A keyboard key is pressed
keyup	onkeyup	A keyboard key is released
load	onload	A web page is loaded
mousedown	onmousedown	One of the mouse buttons is pressed
mousemove	onmousemove	The mouse is moved
mouseout	onmouseout	The mouse is moved off an element
mouseover	onmouseover	The mouse is moved over an element
mouseup	onmouseup	The mouse's button is released
reset	onreset	A form's Reset button is clicked
resize	onresize	An element is resized
submit	onsubmit	A form's Submit button is clicked
unload	onunload	The browser unloads a web page

onLoad, onResize

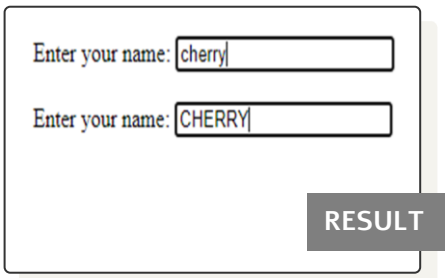
```
<script type = "text/javascript">
   function load(){
                alert("Page Loaded!");}
   function resize(){
                alert("Ouch, that hurts");}
</script>

<body onload="load();" onresize="resize();">
</body>
```



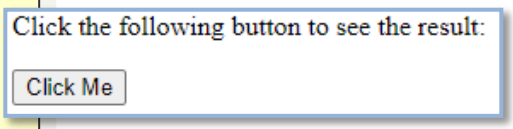
onChange

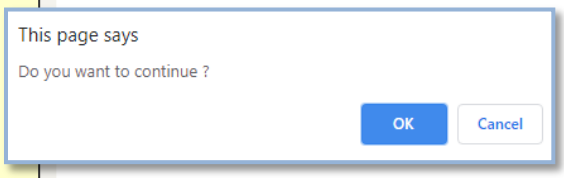
```
<script type = "text/javascript">
  function upperCase(){
  var y=document.getElementById(x).value;
  document.getElementById(x).value=y.toUpperCase();
}
</script>
<body>
   Enter your name: <input type="text" id="fname"
   onchange="upperCase(this.id)" />
</body>
```

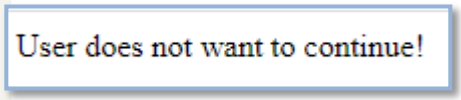


onClick

```
<script type = "text/javascript">
  function getConfirmation() {
    var retVal = confirm("Do you want to continue ?");
    if( retVal == true ) {
          document.write ("User wants to continue!");
     } else {
          document.write ("User does not want to continue!");}
  }
</script>
<body>
     <p>Click the following button to see the result: </p>
     <form>
       <input type = "button" value = "Click Me" onclick =
"getConfirmation();" />
     </form>
 </body>
```







UNIT 6 – Javascript Dialog Boxes

JavaScript supports three important types of dialog boxes. These dialog boxes can be used to raise and alert, or to get confirmation on any input or to have a kind of input from the users. Here we will discuss each dialog box one by one.

The alert Dialog Box

An alert dialog box is mostly used to give a warning message to the users. For example, if one input field requires to enter some text but the user does not provide any input, then as a part of validation, you can use an alert box to give a warning message

```
<script type = "text/javascript">
 function Warn() {
     alert ("This is a warning message!");
     document.write ("This is a warning message!");
 }
</script>
<body>
    <p>Click the following button to see the result: </p>
    <form>
      <input type = "button" value = "Click Me" onclick = "Warn();" />
    </form>
 </body>
```

The confirm Dialog Box

A confirm dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: **OK** and **Cancel**.

If the user clicks on the OK button, the window method confirm**()** will return true. If the user clicks on the Cancel button, then confirm**()** returns false

See example code in onclick section

The prompt Dialog Box

The prompt dialog box is very useful when you want to pop-up a text box to get user input. Thus, it enables you to interact with the user. The user needs to fill in the field and then click OK.

This dialog box is displayed using a method called prompt**()** which takes two parameters: (i) a label which you want to display in the text box and (ii) a default string to display in the text box.

This dialog box has two buttons: **OK** and **Cancel**. If the user clicks the OK button, the window method prompt**()** will return the entered value from the text box. If the user clicks the Cancel button, the window method prompt**()** returns null

```
<body>
  <script type = "text/javascript">
    name = prompt("What is your name?")
    document.write("Your name is " + name + " <br />")
  </script>
</body>
```

RESULT

UNIT 7 – Javascript DOM

Document Object Model

The primary purpose of JavaScript code is to interact with the elements of a HTML page. In a typical bit of JavaScript code we will be reading the content of one or more page elements, doing a computation with those inputs, and then putting the results back into other page elements for display.

To make these interactions possible, JavaScript sets up a ***document object model***, or ***DOM*** for short. In the JavaScript DOM every element in the page is represented by a JavaScript software entity called an ***object***. JavaScript objects store data in ***properties*** and can respond to commands by executing ***methods***. Our entry point to the DOM is a pair of pre-defined DOM variables, ***window*** and ***document***, that give us access to objects representing the window and its contents.

DOM tree shaped structure built out of all of the HTML elements in a page, accessible via JavaScript

A set of JavaScript objects that represent each element on the page

- each tag in a page corresponds to a JavaScript DOM object
- most JS code manipulates elements on an HTML page
- can examine elements' state *e.g.* see whether a box is checked
- can change state *e.g.* insert some new text into a div
- can change styles *e.g.* make a paragraph red

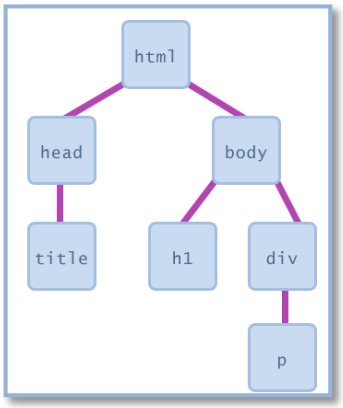
What is a Node?

A simple way to think of the ***DOM*** is in terms of the document tree. Each element within the page is referred to as a ***node***. If you think of the DOM as a tree, each ***node*** is an individual branch that can contain further branches.

- ***Element nodes*** contains an HTML tag
- ***text nodes*** contains text

```
<html>
  <head>
    <title> ... </title>
  </head>
  <body>
    <h1> ... </h1>
    <div>
      <p> ... </p>
    </div>
  </body>
</html>
```

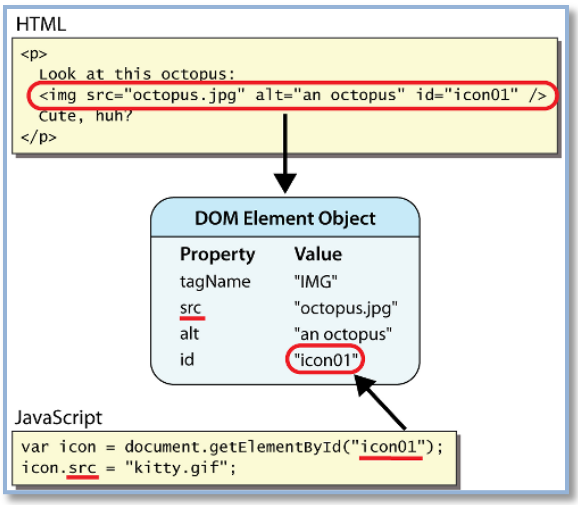
HTML Code



A Document Structure with nodes

DOM Element Objects

Every element on the page has a corresponding DOM object



How JavaScript interacts with DOM Elements Object

Accessing DOM elements

The ***document*** object in the DOM identifies the page itself, and more often than not will serve as the starting point for our DOM crawling. The ***document*** object comes with a number of standard properties and methods for accessing collections of elements.

There are several methods for accessing nodes in the document.

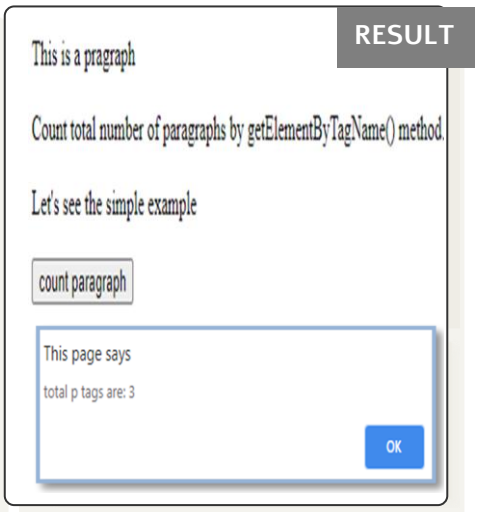
- By element name
- By id attribute value
- By class attribute value

By element tag name

`getElementsByTagName()` method that returns a collection of all elements in the document with the specified tag name

```
<script type="text/javascript">
 function countpara(){
 var totalpara=document.getElementsByTagName("p");
 alert("total p tags are: "+totalpara.length);
}
</script>
<p>This is a pragraph</p>
<p>Here we are going to count total number of paragraphs
    by getElementByTagName() method.</p>
<p>Let's see the simple example</p>
<button onclick="countpara()">count paragraph</button>
```

RESULT



- In this example, it count total number of paragraphs used in the document.
- the `document.getElementsByTagName("p")` method is used to returns the total paragraphs.
- The variable `totalpara` is an array of string and has `.length` property that returns the number or length of an array.

```
<script type="text/javascript">
 function counth2(){
     var totalh2=document.getElementsByTagName("h2");
     document.write("total h2 tags are: "+totalh2.length);  }
 function counth3(){
    var totalh3=document.getElementsByTagName("h3");
    document.write ("total h3 tags are: "+totalh3.length);  }
</script>
 <h2>This is h2 tag</h2>
 <h2>This is h2 tag</h2>
 <h3>This is h3 tag</h3>
 <h3>This is h3 tag</h3>
 <h3>This is h3 tag</h3>
 <button onclick="counth2()">count h2</button>
 <button onclick="counth3()">count h3</button>
```

RESULT

This is h2 tag

This is h2 tag

This is h3 tag

This is h3 tag

This is h3 tag

count h2 | count h3

total h3 tags are: 3

total h2 tags are: 2

By element name

.getElementsByName() method returns all the element of specified name

```
<script type="text/javascript">
  function totalGender(){
    var allgenders=document.getElementsByName("gender");
    document.write ("Total Genders:"+allgenders.length); }
 </script>
<form>
Male:<input type="radio" name="gender" value="male">
Female:<input type="radio" name="gender" value="female">
  <input type="button" onclick="totalGender();" value="Total
Genders">
</form>
```

RESULT

Male: ○ Female: ○ Total Genders

Total Genders:2

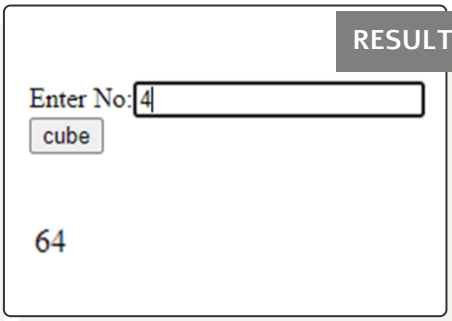
- In this example, it count total number of genders.
- the document.getElementsByName("gender") method is used to get all the genders
- The variable allgenders is an array of string and has .length property that returns the number or length of an array.

By id attribute value

getElementById() This method returns a single element based on that element's ID (the value of its id attribute)

```
<script type="text/javascript">
 function totalGender(){
    var intNum=document.getElementsById("number");
    document.write(intNum * intNum * intNum );    }
</script>
<form>
Enter No:<input type="text" id="number" name="number"
 /><br/>
<input type="button" value="cube" onclick="getcube()"/>
</form>
```

RESULT

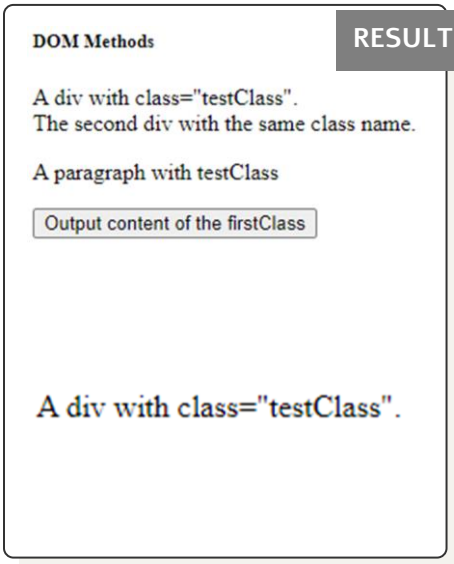


By class attribute value

getElementsByClassName method is used for selecting or getting the elements through their class name value

```
<script type="text/javascript">
 function testFunction(){
   var x=document.getElementByClassName("testClass");
    document.write(x[o].innerHTML);    }
</script>
<body>
<div class="testClass">A div with class="testClass".</div>
<div class="testClass">The second div with the same class
name.</div>
<p class="testClass">A paragraph with testClass</p>
 <form>
<input type="button" value="Output content of the
 firstClass" onclick="testFunction()"/>
</form>
</body>
```

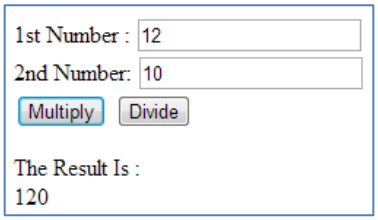
RESULT



ASSESSMENT/ACTIVITIES GUIDE QUESTION/s

Guide Questions

1. What is JavaScript used for?
2. Name one good thing and one bad thing about linking to an external .js file.
3. Describe what this does: for(var i = 0; i <= items.length; i++) { }
4. What does this do? document.getElementById("main").getElementsByTagName("section");
5. Create a web page which prints Fibonacci series from 1 to 10 in JavaScript
6. Write a JavaScript program to determine whether a given year is a leap year
7. Write a JavaScript program to calculate multiplication and division of two numbers (input from user).
   Sample



8. Write a JavaScript program to convert temperatures to and from Celsius, Fahrenheit.
   [Formula : c/5 = (f-32)/9 [where c = temperature in Celsius and f = temperature in Fahrenheit]
   Expected Output :
   60°C is 140 °F
   45°F is 7.222222222222222°C

9. Write a JavaScript for loop that will iterate from 0 to 15. For each iteration, it will check if the current number is odd or even, and display a message to the screen.
   SampleOutput :
   "0 is even"
   "1 is odd"
   "2 is even"

10. Create a webpage with an of Homer Simpson image at the center of the page. Develop a script that prints an alert: “Duh, you are hovering!!” every time the mouse crosses over the image.

- ☐ Add 5 buttons to your webpage: red, yellow, green, black, and silver. Every time you click on one of these buttons the background should take the corresponding color.
- ☐ Add a link with the text: “CLICK ME!”. Develop a function that randomly chooses between the following websites to link your text:
  - ☐ http://slashdot.org/
  - ☐ http://www.thinkgeek.com/
  - ☐ http://despair.com/

LESSON 6

PHP

INTRODUCTION

PHP is the most popular server-side language used to build dynamic websites, and though it is a very extensive language, this class will take it step-by-step.

The stateless web (HTML, CSS and JavaScript) can only do so much without a dynamic language such as PHP to add the ability to interact with the web server.

This lesson will discuss how to make your pages dynamic based upon user interaction, interacting with HTML forms and store and retrieve information.

LEARNING OUTCOMES

At the end of this module, the students are expected to:

- Understand how server-side programming works on the web.
- Learn PHP Basic syntax for variable types and calculations.
- Understand the different control structure in PHP
- Understanding POST and GET in form submission.
- Understand How to receive and process form submission data.

COURSE CONTENTS

UNIT 1 – Introduction to PHP

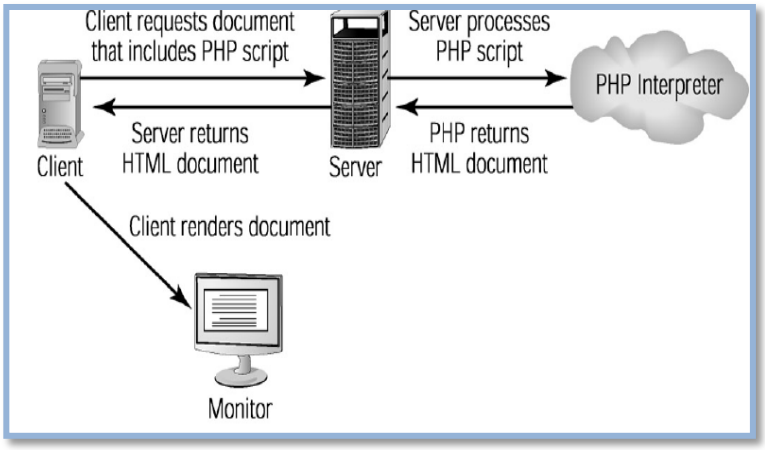
What is PHP?

PHP is a server-side scripting language designed specifically for the web. Within an HTML page, you can embed PHP code that will be executed each time the page is visited. Your PHP code is interpreted at the web server and generates HTML or other output that the visitor will see.

PHP was conceived in 1994 and was originally the work of one man, Rasmus Lerdorf. It was adopted by other talented people and has gone through several major rewrites to bring us the broad, mature product we see today.

PHP is an open-source project, which means you have access to the source code and have the freedom to use, alter, and redistribute it

The home page for PHP is available at http://www.php.net.



PHP on Web Server

PHP Features

- Form processing
- Output / generate various types of data (not just text)
- Database access
  - Allows for various DBs and DB formats
- Object-oriented features
  - Somewhat of a loose hybrid of C++ and Java

PHP's Strengths

Some of PHP's main competitors are Python, Ruby (on Rails or otherwise), Node.js, Perl, Microsoft .NET, and Java. In comparison to these products, PHP has many strengths, including the following:

- Performance
- Scalability
- Interfaces to many different database systems
- Built-in libraries for many common web tasks
- Low cost
- Ease of learning and use
- Strong object-oriented support
- Portability
- Flexibility of development approach
- Availability of source code
- Availability of support and documentation

PHP File

PHP files may contain text, HTML tags and scripts. PHP files are returned to the browser as plain HTML. It has a file extension of ".php", ".php3", or ".phtml"

PHP Installation Prerequisites

- Install Apache and PHP
  - XAMPP (Cross Platform Apache, MySql, PHP, Perl)
  - WAMP (Windows, Apache, MySql, PHP/Perl/Phyton)
  - LAMPP (Linux Apache, MySql, PHP, Perl)
- Choose an appropriate PHP IDE to help you write code faster and more efficiently
  - Sublime Text
  - Visual Studio Code
  - Bluefish
  - Notepad ++

UNIT 2 – PHP Basics

Embedding PHP in HTML

A PHP script can be placed anywhere in the document. Default delimiter syntax opens with <?php and concludes with ?>

Syntax:

```
<?php // PHP code ?>
```

NOTE: The default file extension for PHP files is ".php".
A PHP file normally contains HTML tags, PHP scripting code.

Below is an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

PHP

```
<html>
  <body>
    <?php echo "Hello World!"; ?>
  </body>
</html>
```

OUTPUT

Hello World!

NOTE: PHP statements should end with a semicolon (;)

PHP Case Sensitivity

In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.

In the example below, all three echo statements below are legal (and equal):

```
<?php
ECHO "Hello World! <br/>";
echo "Hello World! <br/>";
EcHo "Hello World! <br/>";
?>
```

NOTE: However; all variable names are case-sensitive.

Commenting Your Code

A comment in PHP code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is looking at the code. In PHP, we use `//` to make a single-line comment or `/*` and `*/` to make a large comment block.

PHP

```
<html>
  <body>
    <?php
    // This is a single-line comment
    # This is also a single-line comment
   /* This is a multiple-lines comment block that
      spans over multiple lines */
    ?>
  </body>
</html>
```

Data Types

Data Types defines the type of data a variable can store. PHP supports the following basic data types:

- Integer—Used for whole numbers
- Float (also called double)—Used for real numbers
- String—Used for strings of characters
- Boolean—Used for true or false values

- Array—Used to store multiple data items
- Object—Used for storing instances of classes

Variables

Variables are used for storing values, such as numbers, strings or functions results, so that they can be used many times in a script. All variables in PHP start with a ($)sign symbol.

Rules in Naming Variables

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with underscore ($my_string), or with capitalization ($myString)

Assigning Value to a Variable

Variables do not have to be explicitly declared in PHP. PHP automatically converts the variable to the correct data type, depending on how they are set. In PHP the variable is declared automatically when you use it.

Assignment by value simply involves copying the value of the assigned expression to the variable assignee. This is the most common type of assignments

```
<?php
$color = “red”;
$number=12;
$sum = 15+”12” //$sum = 27
?>
```

Type Strength

PHP is called a weakly typed or dynamically typed language. PHP automatically converts the variable to the correct data type, depending on how they are set.

```
<?php
$totalqty = 0;
$totalamount=100.00;
?>
```

Since 0, an integer was assigned to $totalqty, this is now an integer variable

Similarly, $totalamount is now of type float

Type Casting

Converting values from one datatype to another is known as ***type casting.*** Variable can be evaluated once as a different type by casting it to another. This is accomplished by placing the intended type in front of the variable to be cast.

Cast Operator	Conversion
(array)	Array
(bool) or (boolen)	Boolean
(int) or (integer)	Integer
(real) or (float) or (double)	Float
(string)	String

- cast an integer as a double

```
$score = (double) 13; // $score = 13.0
```

- Type casting a double to an integer will result in the integer value being rounded down, regardless of the decimal value

```
$score = (int) 14.8; // $score = 14
```

- cast a string datatype to that of an integer. In light of PHP's loosely typed design, it will simply return the integer value unmodified

```
$sentence = "This is a sentence";
echo (int) $sentence; // returns 0
```

- You can also cast a datatype to be a member of an array. The value being cast simply becomes the first element of the array

```
$score = 1114;
$scoreboard = (array) $score;
echo $scoreboard[0]; // Outputs 1114
```

Adapting Datatypes with Type Juggling

Because of PHP's lax attitude toward type definitions, variables are sometimes automatically cast to best fit the circumstances in which they are referenced

```
<?php
$total = 5; // an integer
$count = "15"; // a string
$total += $count; // $total = 20 (an integer)
?>
```

- The outcome is the expected one; $total is assigned 20, converting the $count variable from a string to an integer in the process

```
<?php
$total = "45 fire engines";
$incoming = 10;
$total = $incoming + $total; // $total = 55
?>
```

- The integer value at the beginning of the original $total string is used in the calculation. However, if it begins with anything other than a numerical representation, the value is 0

```
<?php
$total = "1.0";
if ($total) echo "We're in positive territory!";
?>
```

- a string is converted to Boolean type in order to evaluate the if statement

Variable Scope

The term scope refers to the places within a script where a particular variable is visible. The six basic scope rules in PHP are as follows:

- Built-in superglobal variables are visible everywhere within a script
- Constants, once declared, are always visible globally; that is, they can be used inside and outside functions
- Global variables declared in a script are visible throughout that script, but not inside functions.

- Variables inside functions that are declared as global refer to the global variables of the same name
- Variables created inside functions and declared as static are invisible from outside the function but keep their value between one execution of the function and the next
- Variables created inside functions are local to the function and cease to exist when the function terminates.

Declaring and Using Constants

Constant is a value that cannot be modified throughout the execution of a program. It useful when working with values that definitely will not require modification, such as pi (3.141592) or the number of feet in a mile (5,280).

The `define()` function defines a constant by assigning a value to a name.

Syntax:

```
<?php
define(string name, value)
?>
```

PHP

```
<?php
define("PI", 3.141592);

printf("The value of pi is %f", PI);
$pi2 = 2 * PI;
printf("Pi doubled equals %f", $pi2);
?>
```

OUTPUT

```
The value of pi is 3.141592
Pi doubled equals 6.283184
```

Expressions

Expression is a phrase representing a particular action in a program. All expressions consist of at least one operand and one or more operators

```
$a = 5; // assign integer value 5 to the variable $a
$a = "5"; // assign string value "5" to the variable $a
$sum = 50 + $some_int; // assign sum of 50 + $some_int to $sum
$wine = "Zinfandel"; // assign "Zinfandel" to the variable $wine
$inventory++; // increment the variable $inventory by 1
```

Operators

Operators are symbols that you can use to manipulate values and variables by performing an operation on them

Arithmetic Operators

Operator	Name	Example
+	Addition	$a + $b
-	Subtraction	$a - $b
*	Multiplication	$a * $b
/	Division	$a / $b
%	Modulus	$a % $b

Assignment Operators

Assign a data value to a variable using assignment operator **(=)**

The simplest form of assignment operator just assigns some value, while others (known as shortcut assignment operators) perform some other operation before making the assignment

Example	Purpose
$a = 5	Assignment
$a += 5	Addition Assignment
$a *= 5	Multiplication Assignment
$a /= 5	Division Assignment
$a .= 5	Concatenation Assignment

String Operators

PHP's string operators provide a convenient way in which to concatenate strings together. There are two such operators, including the concatenation operator (.) and the concatenation assignment operator (.=)

```
//$a contains the string value "Spaghetti & Meatballs"
$a = "Spaghetti" . "& Meatballs";
$a .= " are delicious."
print $a
```

OUTPUT

Spaghetti& Meatballs are delicious.

Increment and Decrement Operators

The increment (++) and decrement (--) operators present a minor convenience in terms of code clarity, providing shortened means by which you can add 1 to or subtract 1 from the current value of a variable

++$a, $a++	Increment	Increment $a by 1
--$a, $a--	Decrement	Decrement $a by 1

Logical Operators

Logical Operators are another group of operators vital to programming, because they are useful for making decisions in a program. It direct the flow of a program and are used frequently with control structures, such as the if conditional and the while and for loops.

Operator	Name	Use	Result
!	NOT	!$b	Returns true if $b is false and vice versa
&&	AND	$a && $b	Returns true if both $a and $b are true; otherwise false
||	OR	$a || $b	Returns true if either $a or $b or both are true; otherwise false
and	AND	$a and $b	Same as &&, but with lower precedence
or	OR	$a or $b	Same as ||, but with lower precedence
xor	XOR	$a x or $b	Returns true if either $a or $b is true, and false if they are both true or both false.

Equality Operators

Equality operators are used to compare two values, testing for equivalence

Example	Label	Output
$a == $b	Is equal to	True if $a and $b are equivalent
$a != $b	Is not equal to	True if $a is not equal to $b
$a === $b	Is identical to	True if $a and $b are equivalent and $a and $b have the same type

Ternary Operator

The ternary operator (?:) takes the following form:

```
condition? yes-expression : no-expression
```

Allows for simple logical decisions to be made but should be used sparingly as it can lead to very unwieldy and confusing code

```
($a == 12) ? 5 : -1
```

- The first operand is the condition to be evaluated
- The second expression, between the question mark and colon, is evaluated when the condition evaluates to True.
- The final operand is evaluated when the condition is False

Relational Operator

Test the relationship of the two operands, returning a Boolean result

Operator	Name	Use
==	Equals	$a == $b
===	Identical	$a === $b
!=	Not equal	$a != $b
!==	Not identical	$a !== $b
<>	Not equal (comparison operator)	$a <> $b
<	Less than	$a < $b
>	Greater than (comparison operator)	$a > $b
<=	Less than or equal to	$a <= $b
>=	Greater than or equal to	$a >= $b

The PHP echo Statement

The echo() statement is used to output to the screen. It is capable of outputting multiple strings. The echo statement can be used with or without parentheses: echo or echo()

The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

PHP

```
<?php
echo “<h4>PHP is fun</h4>”;
echo ”Hello world! <br/>”;
echo ”I’m about to learn PHP!<br/>”;
echo ”This ”, ”string ”, ”was ”, ”made ”,
”with multiple parameters.”;
?>
```

OUTPUT

PHP is fun

Hello world!
I'm about to learn PHP!
This string was made with multiple parameters.

The following example shows how to output text and variables with the echo statement:

```php
<?php
$txt1 = "Learn PHP";
$txt2 ="Akre";
$x = 5;
$y = 4;
echo "<h2>$txt1</h2>";
echo "Study PHP at $txt2<br>";
echo $x + $y;
?>
```

PHP

Learn PHP OUTPUT

Study PHP at Akre
9

The PHP print Statement

The print() statement outputs data passed to it to the browser. Unlike echo, print can accept only one argument. Unlike echo, print returns a value, which represents whether the print statement succeeded.

The print statement can be used with or without parentheses: print or print()

The following example shows how to output text with the print command (notice that the text can contain HTML markup):

```php
<?php
print "<h4>PHP is fun</h4>";
print "Hello world! <br/>";
print "I'm about to learn PHP!<br/>";
?>
```

PHP

PHP is fun OUTPUT

Hello world!
I'm about to learn PHP!

The following example shows how to output text and variables with the print statement:

```php
<?php
$txt1 = "Learn PHP";
$txt2 ="Akre";
$x = 5;
$y = 4;
print "<h2>$txt1</h2>";
print "Study PHP at $txt2<br>";
print $x + $y;
?>
```

PHP

Learn PHP OUTPUT

Study PHP at Akre
9

The if Statement

The **if** statement is used to execute some code only if a specified condition is true.

Syntax:

```
if (condition) {
    code to be executed if condition is true;
}
```

The if..else Statement

Use the **if....else** statement to execute some code if a condition is true and another code if the condition is false.

Syntax:

```
if (condition) {
    code to be executed if condition is true;
}
else {
    code to be executed if condition is false;
}
```

The if...elseif....else Statement

Use the **if....elseif...else** statement to specify a new condition to test, if the first condition is false

Syntax:

```
if (condition) {
    code to be executed if condition is true;
}
elseif (condition) {
    code to be executed if condition is true
else {
    code to be executed if condition is false;
}
```

The switch Statement

The switch statement is used to perform different actions based on different conditions. Use the switch statement to select one of many blocks of code to be executed.

Syntax:

```
switch (expression) {
  case value1: code to be executed if expression = value1;
       break;
  case label2: code to be executed if expression = value2;
       break;
  default: code to be executed if expression is different from both value1and value2;
  }
```

This is how it works:

- A single expression (most often a variable) is evaluated once.
- The value of the expression is compared with the values for each case in the structure.
- If there is a match, the code associated with that case is executed.
- After a code is executed, break is used to stop the code from running into the next case.
- The default statement is used if none of the cases are true.

The while Loop

The **while** loop executes a block of code as long as the specified condition is true.

Syntax:

```
while (condition is true) {
      code to be executed;
}
```

PHP

```
<?php
$x = 1;
while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

OUTPUT

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

The do...while Loop

The **do...while** loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax:

```
do {
      code to be executed;
} while (condition is true);
```

PHP

```
<?php
$x = 1;
do {
  echo "The number is: $x <br>";
  $x++;
} while($x <= 5);
?>
```

OUTPUT

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

The for Loop

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

Syntax:

```
for (initialization counter; test counter; increment counter) {
      code to be executed;
}
```

initialization counter	Initialize the loop counter value.
test counter	Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
increment counter	Increases the loop counter value

PHP

```
<?php
for ($x = 1; $x <= 5; $x++)  {
 echo "The number is: $x ";
}
?>
```

OUTPUT

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

The foreach Loop

The **foreach** loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax:

```
foreach ($array as $value) {
     code to be executed;
}
```

```php
<?php
  $colors = array("red", "green", "blue", "yellow");

  foreach ($colors as $value)  {
     echo "Value is: $value <br> ";
   }
?>
```

PHP

OUTPUT

```
Value is: red
Value is: green
Value is: blue
Value is: yellow
```

UNIT 4 – PHP Functions

Functions

A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

Creating Functions

- All functions start with the word `function()`
- Name the function - It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number).
- Add a "{" - The function code starts after the opening curly brace.
- Insert the function code.
- Add a "}" - The function is finished by a closing curly brace

Syntax:

```
function function_name {
    code to be executed;
}
```

```php
<?php
function writeMyName(){
        echo "Areen";
        }

echo "My name is ";
writeMyName();
?>
```

PHP

OUTPUT

```
My name is Areen
```

- In the example, it is a simple function that writes my name when it is called

Adding Parameters Functions

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

PHP

```
<?php
function addNum($x,$y){
        echo "the sum of two numbers is ",$x+$y;
 }
 addnum(4,3);
?>
```

OUTPUT

```
the sum of two numbers is 7
```

Functions that Return Values

A ***function*** can return a value after it is done/Use this value in future computation, use like a variable, assign value to a variable

Use the return command to return a value in a function

PHP

```
<?php
function multiplyNum($x,$y){
        return $x * $y;
 }
 echo multiplyNum (4,3);
$a= multiplynum(2,5);
echo "<br>",    $a;
?>
```

OUTPUT

```
12
10
```

UNIT 5 – PHP Arrays

Arrays

An array is a special variable, which can hold more than one value at a time. An array stores multiple values in one single variable

Each item is distinguished by a special identifier known as a ***key***.

Each item consists of two components: ***key*** and a ***value***

The key serves as the lookup facility for retrieving its counterpart, the ***value***.

Keys can be ***numerical*** or ***associative***

In PHP, there are three types of arrays:

- ***Numeric arrays*** - Arrays with a numeric index
- ***Associative arrays*** - Arrays with named keys
- ***Multidimensional arrays*** - Arrays containing one or more arrays

Numeric Arrays

A numeric array stores each element with a numeric ID key.

In this example the ID key is automatically assigned:

```
$names = array("Peter","Quagmire","Joe");
```

In this example the ID key is explicitly assigned and can be used in script:

```
<?php
$names[0] = "Peter";
$names[1] = "Quagmire";
$names[2] = "Joe";
echo $names[1] . " and " . $names[2] .
        " are "  . $names[0] . "'s neighbors";
?>
```

PHP

```
Quagmire and Joe are Peter's neighbors
```

OUTPUT

Associative Array

An ***associative array***, each ID key is associated with a value. When storing data about specific named values, a numerical array is not always the best way to do it. With ***associative arrays*** we can use the values as keys and assign values to them.

In this example the ID key is explicitly assigned

```
$ages = array("Peter"=>32,"Quagmire"=>30,"Joe"=>34);
```

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
echo "Peter is "  $ages['Peter'] . " years old.";
?>
```

PHP

```
Peter is 32 years old
```

OUTPUT

NOTE: You can use count function to return the length of array.
Like count($age)→ 3

Multidimensional Array

A multidimensional array is an array containing one or more arrays.

PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

PHP

```php
<?php
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

OUTPUT

Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.

NOTE: For a two-dimensional array you need two indices to select an element
For a three-dimensional array you need three indices to select an element

Sort Functions For Arrays

sort()	sort arrays in ascending order
rsort()	sort arrays in descending order
asort()	sort associative arrays in ascending order, according to the value
ksort()	sort associative arrays in ascending order, according to the key
arsort()	sort associative arrays in descending order, according to the value
krsort()	sort associative arrays in descending order, according to the key

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

$clength = count($cars);
for($x = 0; $x < $clength; $x++) {
  echo $cars[$x];
  echo "<br>";
}
?>
```

PHP

```
BMW
Toyota
Volvo
```

OUTPUT

UNIT 6 – PHP Form Handling

PHP Form

HTML <form> elements are commonly used to create interfaces that allow users to interact with your dynamic site.

Creating a form on the webpage is accomplished using HTML, while PHP serves as a transport for those values from the webpage to the server and then in further processing those values.

HTML forms are used to pass data to a server via different input controls. All input controls are placed in between <form> and </form>

PHP provides two ***superglobals*** $_GET and $_POST for collecting form-data for processing

Syntax:

```
<form action="PagetoOpen" method="">
   // code to be executed;
</form>
```

In the Syntax

<form>	tag used to create an HTML form
action	Using this attribute, we can specify the name of the file which will collect and handle the form-data.
method	This attribute specify the means of sending the form-data, whether it will be submitted via POST method or GET method

HTML

```
<form action="form.php" method="POST">
   Name: <input type="text" name="name" />
   Email: <input type="text" name="email" />
   <input type="submit" value="Submit" />
</form>
```

PHP $_GET

The $_GET variable is an array of variable names and values sent by the HTTP GET method.

The $_GET variable is used to collect values from a form with method="get". Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and it has limits on the amount of information to send (max. 100 characters).

The start of GET data is indicated with a question mark (?). The name of the field is followed by an equal sign (=) and the value of the field. Each field is then separated by an ampersand (&)

```
<?php
 if($_GET['Submit'] == "Submit") {
  $varMovie = $_GET['Movie'];
  echo "<br /> Your favorite movie is $varMovie";
 }
?>
<form action="myformget.php" method="get"/>
Which is your favorite movie?
<input type="text" name="Movie" maxlength="50"
        value="<?php echo $_GET['Movie'] ;?>">
<input type="submit" name="Submit" value="Submit">
```

OUTPUT


```
http://localhost/sample/myformget.php?Movie=Titanic&Submit=Submit
```

Why use $_GET?

When using the $_GET variable all variable names and values are displayed in the URL. So this method should not be used when sending passwords or other sensitive information. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

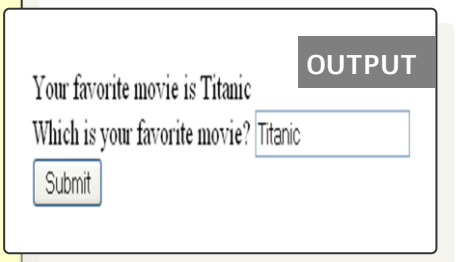
The HTTP GET method is not suitable on large variable values; the value cannot exceed 100 characters.

PHP $_POST

The $_POST variable is an array of variable names and values sent by the HTTP POST method.

The $_POST variable is used to collect values from a form with method="post". Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send

```
<?php
 if($_POST['Submit'] == "Submit")
 {
  $varMovie = $_ POST ['Movie'];
  echo "<br /> Your favorite movie is $varMovie";
 }
?>
<form action="myformPost.php" method="post"/>
Which is your favorite movie?
<input type="text" name="Movie" maxlength="50"
        value="<?php echo $_ POST ['Movie'] ;?>">
<input type="submit" name="Submit" value="Submit">
```



Why use $_POST?

Variables sent with HTTP POST are not shown in the URL
Variables have no length limit
However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

$_SERVER['PHP_SELF'] as a form action

The $_SERVER auto-global array holds various useful server- and request-specific info
The PHP_SELF element of $_SERVER holds the filename of the currently executing script
Supplying $_SERVER['PHP_SELF'] as the action attribute of the form tag makes the form submit to the same page that displayed it
This lets you put the logic to handle the form in the same page as the logic that displays it

```
//movie.php
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
Which is your favorite movie?
<input type="text" name="Movie" maxlength="50"
        value="<?php echo $_POST['Movie'];?>">
<input type="submit" name="Submit" value="Submit">
</form>
<?php
 if($_POST['Submit'] == "Submit")  {
   $varMovie = $_POST['Movie'];
   echo "<br /> Your favorite movie is $varMovie";
 }else
        $_POST['Movie'] = ''
?>
```

Using isset

You can use the isset function on any variable to determine if it has been set or not. You can use this function on the $_POST array to determine if the variable was posted or not.This is often applied to the submit button value, but can be applied to any variable.

```
<form action="isset.php" method="post">
  <p>First name: <input type="text" name="firstname" /></p>
  <p>Last name: <input type="text" name="lastname" /></p>
  <input type="submit" name="submit" value="Submit" />
</form>
<?php
  if(isset($_POST['submit']))  {
    echo("First name: " . $_POST['firstname'] . "<br />\n");
    echo("Last name: " . $_POST['lastname'] . "<br />\n");
  }
?>
```

PHP $_REQUEST

Instead of using GET and POST arrays, you can also use the $_REQUEST array, which will contain the combined contents of the data

The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods

```
<?php
  if(isset($_POST['submit'])) {
    echo("First name: " . $_REQUEST['firstname'] . "<br />\n");
    echo("Last name: " . $_REQUEST['lastname'] . "<br />\n");
  }
?>
<form action="requestphp.php" method="post">
  <p>First name: <input type="text" name="firstname" /></p>
  <p>Last name: <input type="text" name="firstname" /></p>
  <input type="submit" name="submit" value="Submit" />
</form>
```

ASSESSMENT/ACTIVITIES GUIDE QUESTION/s

Guide Questions

1. What are the characteristics of PHP variables?
2. Define the different data types of PHP variables?
3. What are the differences between PHP constants and variables?
4. What is the purpose of break and continue statement?
5. What are the different types of array in PHP? Explain each.
6. What is the difference between GET and POST?
7. Write a PHP program to check which number nearest to the value 100 among three given integers. Return 0 if the three numbers are equal.
8. Write a PHP program to compute the sum of the elements of an given array of integers.
9. Write a PHP program to count even number of elements in a given array of integers.
10. Write a function that prompts the user to input a number, and then gives them 5 chances to guess the square of the number.
11. Write a function that takes 3 parameters, price, tax rate, balance, and prints yes in an alert box if someone has adequate balance to purchase an item at the specified price with the given tax rate. Otherwise, print out no in an alert box.
12. Write a function tip(bill) that will calculate and return the amount that should be left as a tip for a restaurant bill when passed the parameter bill. The tip should be 20% of the total before a tip is added

LESSON 7

MYSQL

INTRODUCTION

This lesson discuss how to interact with MySQL databases by establishing a connection, making queries, manipulating the results of those queries, and closing the database connection.

LEARNING OUTCOMES

At the end of this module, the students are expected to:

1. Learn how to establish connection, making queries and manipulating the result of queries in MySQL
2. Learn how to create database and tables through programming codes.
3. Learn how to insert, update, and delete records from database
4. Connect PHP to MySQL

UNIT 1 – Introduction to MySQL

MySQL

MySQL is a Relational Database Management System ("RDBMS"). It is used by most modern websites and web-based services as a convenient and fast-access storage and retrieval solution for large volumes of data. A simple example of items which might be stored in a ***MySQL*** database would be a site-registered user's name with associated password (encrypted for security), the user registration date, and number of times visited, etc.

MySQL can also be accessed using many tools. It can be easily communicated with via PHP (PHP Hypertext Preprocessor), a scripting language whose primary focus is to manipulate HTML for a webpage on the server before it is delivered to a client's machine. A user can submit queries to a database via PHP, allowing insertion, retrieval and manipulation of information into/from the database.

Creating a Database

To create a new database, issue the CREATE DATABASE command:

Syntax:

```
CREATE DATABASE [IF NOT EXISTS] <database name>
[[DEFAULT] CHARACTER SET <character set name>]
[[DEFAULT] COLLATE <collation name>]
```

CREATE DATABASE	keyword for creating new database
database_name	name of new database
IF NOT EXISTS	If a database with the same name exists, no new database is created
CHARACTER SET	character set to use for a new database
COLLATE	collation to use, a named sorting order used for a specified character set

```
CREATE DATABASE BookSales;
```

- Created a database named BookSales. The database uses the default character set and collation because you specified neither

```
CREATE DATABASE BookSales
DEFAULT CHARACTER SET latin1
DEFAULT COLLATE latin1_bin;
```

- the CHARACTER SET clause specifies the latin1 character set, and the COLLATE clause specifies the latin1_bin collation

Deleting Databases

To delete a database, issue the DROP DATABASE command:

Syntax:

```
DROP DATABASE [IF EXISTS] <database name>
```

database_name	name of the database to be drop
IF EXISTS	If you specify this clause and a database with that name doesn't exist, you receive a warning message rather than an error

```
DROP DATABASE BookSales;
```

- removes the BookSales database from the system
- When you remove a database, you also remove the tables in that database and any data contained in the table

Data Types

Data type is a constraint placed on a particular column to limit the type of values that you can store in that column. MySQL supports three categories of data types:

- numeric data type
- string data type
- data/time data type

MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

Creating Tables

To create a table, use the CREATE TABLE command:

Syntax:

```
CREATE TABLE [IF NOT EXISTS] <table_name>
        ( column_name datatype
              [NULL | NOT NULL]
      [ DEFAULT constant_expression ]
      [ AUTO_INCREMENT]
      [ , . . . n]
      [PRIMARY KEY]
      [FOREIGN KEY]
  ) ;
```

CREATE TABLE	keyword for creating new table
table_name	name of new table
column_name	Required. Name of the column
data_type	Data type of the column including the argument (length, precision, scale) if applicable
[NOT NULL | NULL]	Optional. Indicates whether null values are allowed in the column. Default is NULL
DEFAULT	Optional. The value you want to be used for the column for any row that is inserted without explicitly supplying a value for that particular column
AUTO_INCREMENT	Optional. Indicates that the column is will auto increment or numbers be generated automatically. Can be used in column configured with an integer data type and the NOT NULL option
[PRIMARY KEY]	assign the column as Primary Key (column in a table that uniquely identify each row in that table)
[FOREIGN KEY]	assign the column as Foreign Key references to other column of a table

```
CREATE TABLE IF NOT EXISTS Categories
 ( CategoryID int NOT NULL
       AUTO_INCREMENT PRIMARY KEY,
   CategoryName VARCHAR(20) NOT NULL,
   Description VARCHAR(20) NULL
  ) ;
```

Deleting a Table

To delete a table, use the DROP TABLE command:

Syntax:

```
DROP TABLE <table name>
```

```
DROP TABLE classics
```

- Drop the Classic table from the database

UNIT 2 – Manipulating Data

The INSERT Statement

Add new rows to a table by using the INSERT statement.

Syntax:

```
INSERT INTO table [(column [,column...])]
VALUES                  (value [, value...])
```

INSERT INTO	commands for inserting data
`table_name`	name of the table where the new row will be added
column	specifies the columns to be updated in the new MySQL row
VALUES (value_1,value_2,...)	specifies the values to be added into the new row

```
INSERT INTO Categories
  (CategoryName, Description)
VALUES
  ('Cold Cuts', 'Ham')
```

- This example would insert one record into the *Categories* table. This new record would have a *category_name* of 'Cold Cuts' and Description as 'Ham'.

The UPDATE Statement

The UPDATE statement in SQL is used to update the data of an existing table in database. You can update single columns as well as multiple columns.

Syntax:

```
UPDATE table
SET column1=value1 [,col2=val2..]
[WHERE conditions]
```

UPDATE	command to update existing record
column	the columns to be updated
value	the new value to assign to the column

`WHERE condition`	Optional. The conditions that must be met for the update to execute.

NOTE: Be careful when updating records in a table! Notice the `WHERE` clause in the `UPDATE` statement. The `WHERE` clause specifies which record(s) that should be updated. If you omit the `WHERE` clause, all records in the table will be updated!

- This example updates the Contact Name and City for a Customer whose ID number is 1 in the Customers table.

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt',
        City= 'Frankfurt'
WHERE CustomerID = 1
```

The DELETE Statement

The `DELETE` statement is used to delete a single record or multiple records from a table in SQL Server.

Syntax:

```
DELETE [FROM] table
[WHERE conditions]
```

`DELETE`	command to delete existing record
`table`	the table you wish to delete the record
`WHERE condition`	Optional. The conditions that must be met for the DELETE to execute

NOTE: You do not need to list fields in the SQL Server `DELETE` statement since you are deleting the entire row from the table.

```
DELETE FROM Customers
WHERE CustomerID = 1
```

- This example delete the Customer whose ID number is 1 in the Customers table.

```
DELETE FROM Customers
```

- This example will delete all records in the Customers table.

UNIT 3 – Queries

The SELECT Statement

The SELECT statement is used to query data from tables. The retrieved rows are selected from one or more table. Such a result table can be used as the basis of a report, for example

Syntax:

```
SELECT [DISTINCT] {*, column [alias],...}
FROM tableName
```

In the syntax:

SELECT	is a list of one or more columns
DISTINCT	suppresses duplicates.
*	selects all columns
column	selects the named column.
alias	gives selected columns different headings.
FROM tablename	specifies the table containing the columns.

The complete syntax of the SELECT statement is:

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_name
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name' export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

In general, clauses used must be given in exactly the order shown in the syntax description. For example, a HAVING clause must come after any GROUP BY clause and before any ORDER BY clause.

The WHERE Clause

In the WHERE clause, a condition is used to select rows from a table. These selected rows form the intermediate result of the WHERE clause. The WHERE clause acts as a kind of filter.

Syntax:

```
SELECT select_expr [, select_expr ...]
FROM table
[WHERE where_condition]
```

Comparisons and Conditions

We set conditions within the WHERE clause. The condition could be an expression, for example, 83 or 15 * 100 as already discussed. Alternatively, it could be a comparison or relation operator with another value, for example =100).

Syntax:

```
WHERE column_name operator expression_value
```

The value of the "column_name" is compared with the value of the expression. The result will be true, false, or unknown

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>,!=	Not equal to

Comparison Operator

Multiple conditions can be combined using the logical operators

Syntax:

```
WHERE column_name1 operator value1
      AND column_name1 operator value1
```

Operators	Meaning
AND	Returns TRUE if both component conditions are TRUE
OR	Returns TRUE if either component condition is TRUE
NOT	Returns TRUE if the following condition is FALSE

Logical Operator

Date and Time Comparisons

To compare date or time, you need to specify the date or time in two single quotes as you do for strings. Use the same format as specified in the database.

If the date is saved in a database in YYYY-DD-MM format (which represents 4 digits year – two digits day – two digits month), then you need to compare this date using the same format

```
WHERE column_name operator '1995-25-07'
```

UNIT 3 – Advanced Queries

Sorting Data – ORDER BY Clause

To sort the result of a query, MySQL uses an ORDER BY clause

Syntax:

```
[ORDER BY {col_name | position} [ASC | DESC]
```

Columns selected for output can be referred to in ORDER BY clauses using column names, column aliases, or column positions. Column positions are integers and begin with 1.

To sort in reverse order, add the DESC (descending) keyword to the name of the column in the ORDER BY clause that you are sorting by. The default is ascending order; this can be specified explicitly using the ASC keyword.

Subquery (inner SELECT)

A subquery is a SELECT statement within another statement. In other words, a table expression can be called from within another table expression. The called table expression is a subquery

Syntax:

```
SELECT select_expr
 FROM table1
WHERE column_name =( SELECT select_expr
                     FROM table1
                     WHERE ...)
```

Since we used an equal sign for the inner subquery so far, this means that the inner query should only return one record.

The IN Operator

The use of the IN operator in a SELECT statement makes multiple comparisons easier. The condition with the IN operator has two forms. The first form is when comparing a field with a list of values separated by commas

```
SELECT * FROM table1
WHERE column1='value1'
      OR column1='value2'
      OR column1='value3'
```

Instead, use the IN operator to make the SELECT statement shorter and easier to read:

```
SELECT * FROM table1
WHERE column1 IN ('value1' , 'value2', 'value3')
```

The BETWEEN Operator

SQL supports a special operator that determines whether a value occurs within a given range of values.

Syntax:

```
WHERE column_name1 BETWEEN value1 AND value2
```

UNIT 4 – Connecting to MySQL

Establish MySQL Connection

MySQL database server can contain many databases, each of which can contain many tables

Connecting to the server via PHP:

```
$conn = mysqli_connect($servername, $username, $password);
```

$conn	is the database connection resource variable.
mysqli_connect(...)	is the function for php database connection
$server_name	is the name or IP address of the server hosting MySQL server.
$user_name	is a valid user name in MySQL server.
$password	is a valid password associated with a user name in MySQL server.

Selecting a Database

Once connected to a DBMS, we can select a database using mysqli_select_db

```
mysqli_select_db($connection ,$databasename);
```

mysqli_connect()	the database selection function that returns either true or false
$database_name	is the name of the database

Execute SQL Queries

The mysqli_query function is used to execute SQL queries.

The function can be used to execute the following query types;

- Insert
- Select
- Update
- delete

```
mysqli_query($connection ,$query);
```

mysqli_query(...) is the function that executes the SQL queries.

$query is the SQL query to be executed

PHP mysqli_fetch_array function

The mysqli_fetch_array function is used fetch row arrays from a query result set.

```
mysqli_fetch_array ($result);
```

mysqli_fetch_array(...) the function for fetching row arrays

$result the result returned by the mysqli_query function.

PHP mysqli_num_rows function

The mysqli_num_rows function is used to get the number of rows returned from a select query.

```
mysqli_num_rows ($result);
```

mysqli_num_rows (...) the row count function

$result the result returned by the mysqli_query function.

PHP mysqli_close function

The mysqli_close function is used to close an open database connection.

```
mysqli_close ($connection);
```

mysqli_close(...) is the PHP function that closes a database connection

```
//connect to MySQL server if (!$dbh)
$dbh = mysqli_connect('localhost', 'root', 'melody');

//if connection failed output error message
if (!$dbh)
        die("Unable to connect to MySQL: " . mysqli_error());

//connect to database)
 if (!mysqli_select_db($dbh,'my_personal_contacts'))
        //if selection fails output error message
        die("Unable to select database: " . mysqli_error());

//SQL select query
 $sql_stmt = "SELECT * FROM my_contacts";

//execute SQL statement
$result = mysqli_query($dbh,$sql_stmt);

 if (!$result)
        //output error message if query execution failed
         die("Database access failed: " . mysqli_error());

         // get number of rows returned
        $rows = mysqli_num_rows($result);

        if ($rows) {
         while ($row = mysqli_fetch_array($result)) {
                echo 'ID: ' . $row['id'] . '<br>';
                echo 'Full Names: ' . $row['full_names'] . '<br>';
                echo 'Gender: ' . $row['gender'] . '<br>';
                echo 'Contact No: ' . $row['contact_no'] . '<br>';
                echo 'Email: ' . $row['email'] . '<br>';
                echo 'City: ' . $row['city'] . '<br>';
                echo 'Country: ' . $row['country'] . '<br><br>';
        }
}
mysqli_close($dbh); //close the database connection
```

ASSESSMENT/ACTIVITIES GUIDE QUESTION/s

Guide Questions

1. Create the following tables (relations) and write an insert statement for each record

supplier (SupplierNo, SupplierName, City):
<S1, Smith, Laramie>
<S2, Jones, Chicago>
<S3, Blake, Chicago>
<S4, Clark, Boston>
<S5, Adams, Boston>

item (ItemNo, ItemName):
<I1, Nut>
<I2, Bolt>
<I3, Cog>
<I4, Cam>

supplied (SupplierNo, ItemNo, Quantity):
<S1, I1, 400>
<S1, I2, 300>
<S1, I3, 500>
<S2, I1, 100>
<S2, I2, 200>
<S3, I3, 300>
<S4, I1, 100>
<S4, I3, 100>

2. This question refers to a table named president with schema as follows:

Field	Type	Null	Key	Default	Extra
last_name	varchar(15)				
first_name	varchar(15)				
state	char(2)				
city	varchar(20)				
birth	date			0000-00-00	
death	date	YES		NULL	

   a. Write an SQL query that returns the full names of presidents who are still alive.
   b. Write an SQL query that finds the state in which the greatest number of presidents have been born
   c. Can you write a query that finds the states in which no presidents have been born? Explain your answer.
   d. Write an SQL query that finds the full name of every president with the same last name as some other president.

3. Write the command in PHP that will connect to database named myDB that resides in 192.168.5.100 using the username root and the password dbserv

LESSON 8

SESSION MANAGEMENT

INTRODUCTION

HTTP is stateless – it does not keep track of the client between requests. It remembers nothing about previous transfers. But sometimes, an information are need to be tracked. This lesson will discuss the solution on this issue, which are the cookies and session

LEARNING OUTCOMES

At the end of this module, the students are expected to:

1. Learn how to use PHP session with web pages
2. Understand how data store to cookies

UNIT 1 – Introduction to Cookies

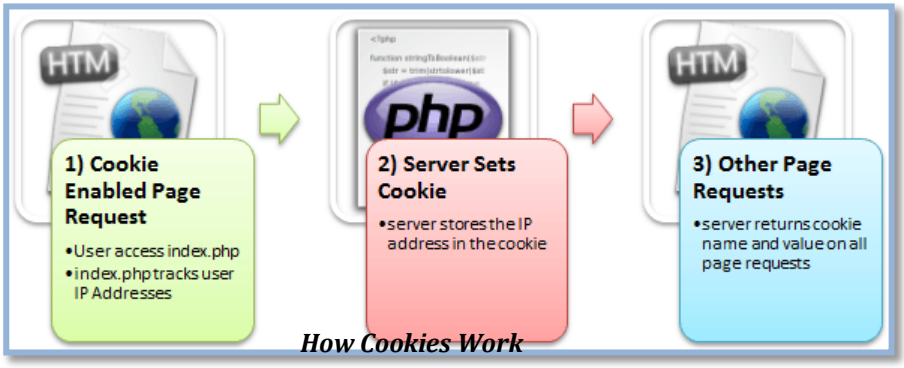
Cookies

A cookie is a small file with the maximum size of 4KB that the web server stores on the client computer.

Once a cookie has been set, all page requests that follow return the cookie name and value.A cookie can only be read from the domain.

A cookie created by a user can only be visible to them. Other users cannot see its value.

Most web browsers have options for disabling cookies, third party cookies or both. If this is the case then PHP responds by passing the cookie token in the URL.



How Cookies Work

Here,

1) A user requests for a page that stores cookies

2) The server sets the cookie on the user's computer

3) Other page requests from the user will return the cookie name and value

Why and when to use Cookies?

Http is a stateless protocol; cookies allow us to track the state of the application using small files stored on the user's computer.

The path were the cookies are stored depends on the browser.

Personalizing the user experience – this is achieved by allowing users to select their preferences.

The page requested that follow are personalized based on the set preferences in the cookies.

Tracking the pages visited by a user

Creating Cookies

```
setcookie(cookie_name, cookie_value, [expiry_time],
        [cookie_path], [domain], [secure], [httponly]);
```

Setcookie	PHP function used to create the cookie.
cookie_name	name of the cookie that the server will use when retrieving its value from the $_COOKIE array variable. It's mandatory.
cookie_value	value of the cookie and its mandatory
[expiry_time]	optional; it can be used to set the expiry time for the cookie such as 1 hour. The time is set using the PHP time() functions plus or minus a number of seconds greater than 0 i.e. time() + 3600 for 1 hour.
[cookie_path]	Optional; it can be used to set the cookie path on the server. The forward slash "/" means that the cookie will be made available on the entire domain. Sub directories limit the cookie access to the subdomain.
[domain]	Optional, it can be used to define the cookie access hierarchy
[secure]	Optional, the default is false. It is used to determine whether the cookie is sent via https if it is set to true or http if it is set to false.

[Httponly] Optional. If it is set to true, then only client side scripting languages

```
setcookie("user_name", "99", time()+ 60,'/'); // expires after 60 seconds
echo 'the cookie has been set for 60 seconds';
```

OUTPUT

```
the cookie has been set for 60 seconds
```

Reading Cookies

To access a cookie received from a client, use the PHP $_COOKIE superglobal array

```
foreach ($_COOKIE as $key=>$val) {
   print $key . " => " . $val . "<br/>";}
```

Each key in the array represents a cookie - the key name is the cookie name.

Delete Cookies

If you want to destroy a cookie before its expiry time, then you set the expiry time to a time that has already passed.

Set the cookie with its name only:

```
setcookie(cookie_name);
```

UNIT 2 – Introduction to Session

Session

A ***session*** is a global variable stored on the server. Each session is assigned a unique id which is used to retrieve stored values. Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL

Sessions have the capacity to store relatively large data compared to cookies.The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.

Just like the $_COOKIE array variable, session variables are stored in the $_SESSION array variable. Just like cookies, the session must be started before any HTML tags.

Why and when to use Sessions?

- You want to store important information such as the user id more securely on the server where malicious users cannot temper with them.
- You want to pass values from one page to another.
- You want the alternative to cookies on browsers that do not support cookies.
- You want to store global variables in an efficient and more secure way compared to passing them in the URL
- You are developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

Creating a Session

In order to create a session, you must first call the PHP session_start function

```
<?php   session_start(); ?>
```

session_start() This tells PHP that a session is requested.

How to Get a Session Id

The server creates a unique number for every new session. If you want to get a session id, you can use the session_id function. A session ID is then allocated at the server end.

```
<?php   session_start();
        echo session_id();
?>
```

session ID looks like: **sess_f12347812374681237 68asjkhfa7891234g**

Session Variables

Once a session is started, the $_SESSION super-global array is initialized with the corresponding session information. By default, it's initialized with a blank array, and you can store more information by using a key-value pair.

To create a session variable, you simply set an element in this array

```
<?php
   $_SESSION['session_variable1'] = value1;
   $_SESSION['session_variable2'] = value2;
?>
```

```
<?php
   session_start();
   $_SESSION['msg'] = "Hello World!";
   echo 'The content of '.$_SESSION['msg'].' is '
.$_SESSION['msg'].'<br/>';
?>
```

OUTPUT

```
The content of  $_SESSION['msg'] is Hello World!
```

Destroy a Session

The session_destroy function deletes everything that's stored in the current session. Having said that, it doesn't unset global variables associated with session or unset the session cookie.

```
<?php   session_destroy(); ?>
```

ASSESSMENT/ACTIVITIES GUIDE QUESTION/s

Guide Questions

1. Why HTTP is stateless?
2. What is the difference between cookies and sessions?
3. True or False? Explain
   Once a Web server returns a cookie to a browser, the cookie will be included in all future requests from the browser to the same server
4. Write a script which creates and retrieves Cookies information.
5. Using session, write a PHP script that will count the number of visitor that logins in a web app. Output the total number of visits.

References:

1. Thomson, Laura and Welling, Luke. PHP and MySQL Web Development 5th edition, Addison-Wesley, 2016
2. Mendez, Michael, The Missing Link An Introduction to Web Development and Programming, 2014
3. Robbins, Jennifer, Learning Web Design 4th edition, O'Reilly, 2012
4. Ducket, John, HTML and CSS, 2011, Wiley, 2011
5. https://www.tutorialspoint.com/javascript/
6. www.w3resource.com