Chapter One

Introduction to Internet Technology and Web Design

1.1. Overview of Internet

What is Internet?

The Internet is a worldwide system of interconnected computer networks. The computers and computer networks exchange information using TCP/IP (Transmission Control Protocol/Internet Protocol) to communicate with each other. The computers are connected via the telecommunications networks, and the Internet can be used for e-mailing, transferring files and accessing information on the World Wide Web.

Advantages

Here, we will discuss some of the advantages of Internet:

- Social Networking
- Education and Technology
- Entertainment
- Online Services

Internet allows us to communicate with the people sitting at remote locations. There are various apps available on the web that uses Internet as a medium for communication. One can find various social networking sites such as: Facebook, Twitter, Yahoo, Google+, Flickr, Orkut etc

One can surf for any kind of information over the internet. Information regarding various topics such as Technology, Health & Science, Social Studies, Geographical Information, Information Technology, Products etc can be surfed with help of a search engine.

Apart from communication and source of information, internet also serves a medium for entertainment. Following are the various modes for entertainment over internet. These are Online Television, Online Games, Songs, Videos and Social Networking Apps.

Internet allows us to use many services like Internet Banking, Matrimonial Services, Online Shopping, Online Ticket Booking, Online Bill Payment, Data Sharing & E-mail.

Internet provides concept of **electronic commerce** that allows the business deals to be conducted on electronic systems

Disadvantages

However, Internet has proved to be a powerful source of information in almost every field, yet there exist many disadvantages discussed below:

- **Threat to Personal Information**: There are always chances to lose personal information such as name, address, credit card number. Therefore, one should be very careful while sharing such information. One should use credit cards only through authenticated sites.
- **Spamming:** it corresponds to the unwanted e-mails in bulk. These e-mails serve no purpose and lead to obstruction of entire system.
- **Virus attack**: Viruses can easily be spread to the computers connected to internet. Such virus attacks may cause your system to crash or your important data may get deleted.

1.1.1. The World Wide Web (WWW)

WWW is also known as W3. It offers a way to access documents spread over the several servers over the internet. These documents may contain texts, graphics, audio, video, hyperlinks. The hyperlinks allow the users to navigate between the documents.

The World Wide Web is a system of Internet servers that use HTTP (Hypertext Transfer Protocol) to transfer documents formatted in HTML (Hypertext Mark-up Language). These are viewed by using software for web browsers such as Netscape, Safari, Google Chrome and Internet Explorer. Hypertext enables a document to be connected to other documents on the web through hyperlinks. It is possible to move from one document to another by using hyperlinked text found within web pages.

1.1.2. Browser and Web Server

Web Browser is application software that allows us to view and explore information on the web. User can request for any web page by just entering a URL into address bar.

Web browser can show text, audio, video, animation and more. It is the responsibility of a web browser to interpret text and commands contained in the web page.

Browser	Vendor
Internet Explorer	Microsoft
Google Chrome	Google
Mozilla Firefox	Mozilla
Netscape Navigator	Netscape Communications Corp.
Opera	Opera Software
Safari	Apple
Sea Monkey	Mozilla Foundation
K-meleon	K-meleon

Web Server

Web server is a computer where the web content is stored. Basically web server is used to host the web sites/systems but there exists other web servers also such as gaming, storage, FTP, email etc.

Web site is collection of web pages while web server is software that responds to the request for web resources.

Web Server Working

Web server respond to the client request in either of the following two ways:

- Sending the file to the client associated with the requested URL.
- Generating response by invoking a script and communicating with database

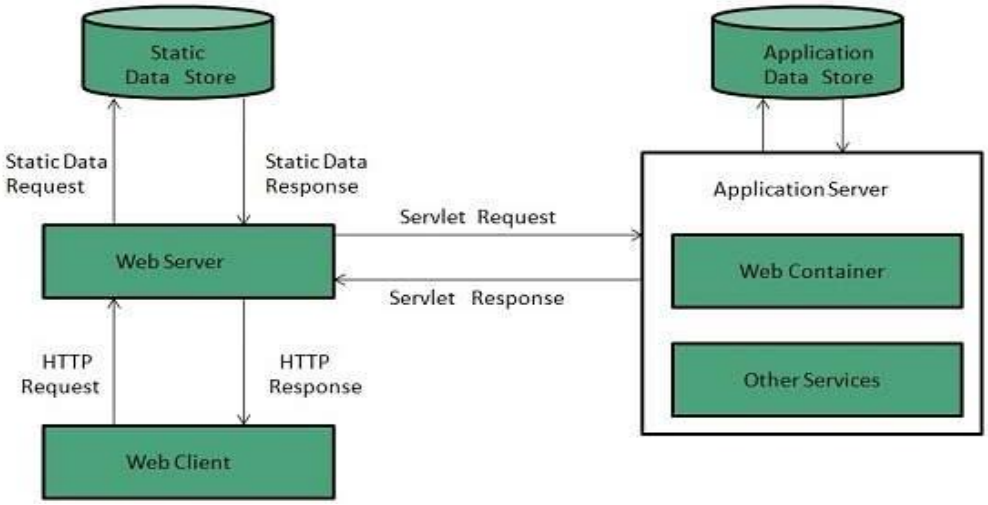


Figure 1.1.2 Client Server Architecture

1.1.3. Document(media) Types

A **media type** (formerly known as **MIME**(*Multipurpose Internet Mail Extensions*) **type**) is a two-part identifier for file formats and format contents transmitted on the Internet.

A media type consists of a *type* and a *subtype*, which is further structured into a *tree*. A media type can optionally define a *suffix* and *parameters*:

```
type "/" [tree "."] subtype ["+" suffix] *[";" parameter]
```

The currently registered types are: `application`, `audio`, `example`, `font`, `image`, `message`, `model`, `multipart`, `text` and `video`.

As an example, an HTML file might be designated `text/html; charset=UTF-8`. In this example, `text` is the type, `html` is the subtype, and `charset=UTF-8` is an optional parameter indicating the character encoding.

A subtype typically consists of a media format, but it may or must also contain other content, such as a tree prefix, producer, product or suffix, according to the different rules in registration trees.

Types, subtypes, and parameter names are case-insensitive. Parameter values are usually case-sensitive, but may be interpreted in a case-insensitive fashion depending on the intended use.

Common examples

- `application/javascript`
- `application/xml`
- `application/zip`
- `application/pdf`
- `application/sql`
- `application/msword` (.doc)
- `application/vnd.openxmlformats-officedocument.wordprocessingml.document`(.docx)
- `application/vnd.ms-excel` (.xls)
- `application/vnd.openxmlformats-officedocument.spreadsheetml.sheet` (.xlsx)
- `application/vnd.ms-powerpoint` (.ppt)
- `application/vnd.openxmlformats-officedocument.presentationml.presentation` (.pptx)
- `application/vnd.oasis.opendocument.text` (.odt)
- `audio/mpeg`
- `audio/ogg`
- `multipart/form-data`
- `text/css`
- `text/html`
- `text/xml`
- `text/csv`
- `text/plain`
- `image/png`
- `image/jpeg`
- `image/gif`

1.1.4. Uniform Resource Locators (URLs)

Uniform Resource Locator (URL) refers to a web address which uniquely identifies a document over the internet. This document can be a web page, image, audio, video or anything else present on the web. For example: www.dmu.edu.et**/index.html**.

URL Types

There are two forms of URL as listed:

- Absolute URL
- Relative URL

Absolute URL

Absolute URL is a complete address of a resource on the web. This completed address comprises of protocol used, server name, path name and file name.

For example http:// www.dmu.edu.et/index.htm. where:

- **http** is the protocol.
- **dmu.edu.et** is the server name.
- **index.htm** is the file name.

The protocol part tells the web browser how to handle the file. Similarly we have some other protocols also that can be used to create URL are: FTP, https, Gopher, mailto, news

Relative URL

Relative URL is a partial address of a webpage. Unlike absolute URL, the protocol and server part are omitted from relative URL. Relative URLs are used for internal links i.e. to create links to file that are part of same website as the WebPages on which you are placing the link. For example, to link an image on dmu.edu.et/bids/bids.pdf, we can use the relative URL which can take the form like **/ bids/bids.pdf**.

Difference between Absolute and Relative URL

Absolute URL	Relative URL
• Used to link web pages on different websites • Difficult to manage. • Changes when the server name or directory name changes • Take time to access	• Used to link web pages within the same website. • Easy to Manage • Remains same even of we change the server name or directory name. • Comparatively faster to access.

1.1.5. Domain Name System (DNS)

DNS is an application layer protocol used to resolve hostnames to IP addresses. Although a host can be accessed by using only its IP address, DNS makes your life easier by using domain names. For example, you can access the Google website by typing http://208.117.229.214 in your browser, but it is much easier to type http://www.google.com.

Each host that wants to use DNS needs to have a DNS server configured. When you type a URL in your browser (e.g. http://www.google.com) , your host will query the DNS server for the IP address of www.google.com. The DNs server will resolve the query and send the answer back to the host. The host will then be able to establish a connection to http://www.google.com.

1.2. Communication Protocols

In telecommunication, a **communication protocol** is a system of rules that allow two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both.

Multiple protocols often describe different aspects of a single communication. A group of protocols designed to work together are known as a protocol suite; when implemented in software they are a protocol stack.

1.2.1. TCP/ IP, HTTP, FTP and Email Protocols

TCP

TCP is a reliable, guaranteed-delivery protocol. TCP specifies the methods hosts use to acknowledge the receipt of packets, and requires the source host to resend packets that are not acknowledged. TCP protocols also govern the exchange of messages between the source and destination hosts to create a communication session. TCP is often compared to a pipeline, or a persistent connection, between hosts. Because of this, TCP is referred to as a connection-oriented protocol.

TCP requires overhead to keep track of the individual conversations between source and destination hosts and to process acknowledgements and retransmissions. In some cases, the delays caused by this overhead cannot be tolerated by the application. These applications are better suited to UDP.

UDP

UDP is a very simple, connectionless protocol. It has the advantage of providing for low overhead data delivery. Because UDP is a "best effort" Transport Layer protocol, UDP datagrams may arrive at the destination out of order, or may even be lost all together. UDP does not provide guaranteed data delivery or flow control. Applications that use UDP can tolerate small amounts of missing data. An example of a UDP application is Internet radio. If a piece of data is not delivered, there may only be a minor effect on the quality of the broadcast.

TCP Vs UDP

UDP is a very simple protocol. Because it is not connection-oriented and does not provide the sophisticated retransmission, sequencing, and flow control mechanisms of TCP, UDP has a much lower overhead.

UDP is often referred to as an unreliable delivery protocol; because there is no guarantee that a message has been received by the destination host. This does not mean that applications that use UDP are unreliable. It simply means that these functions are not provided by the Transport Layer protocol and must be implemented elsewhere if required.

Although the total amount of UDP traffic found on a typical network is often relatively low, key Application Layer protocols that use UDP include:

- Domain Name System (DNS)
- Simple Network Management Protocol (SNMP)
- Dynamic Host Configuration Protocol (DHCP)
- Routing Information Protocol (RIP)
- Trivial File Transfer Protocol (TFTP)
- Online games

those app uses UDP

HTTP and HTTPs

The Hypertext Transfer Protocol (HTTP), one of the protocols in the TCP/IP suite, was originally developed to enable the retrieval of HTML formatted web pages. It is now used for distributed, collaborative information sharing. The HTTP protocol has evolved through multiple versions. The version currently used by most ISP (Internet Service Provider) to provide web-hosting services is HTTP version 1.1. Unlike earlier versions, this version enables a single web server to host multiple web sites. It also permits persistent connections, so that multiple

request and response messages can use the same connection, reducing the time it takes to initiate new TCP sessions.

HTTP specifies a request/response protocol. When a client, typically a web browser, sends a request message to a server, the HTTP protocol defines the message types the client uses to request the web page. The HTTP protocol also defines the message types the server uses to respond.

Although it is remarkably flexible, HTTP is not a secure protocol. The request messages send information to the server in plain text that can be intercepted and read. Similarly, the server responses, typically HTML pages, are also sent unencrypted.

Some HTTP commands (your browser sends these internally)

- GET resource -- requests data from a specified resource
- POST resource -- submits data to be processed to a specified resource
- PUT resource -- uploads a representation of the specified URL
- DELETE resource -- deletes the specified resource

HTTP Status Codes HTTP command fi HTTp status code waan ijoodha

When a request is made by the browser, a response is sent back by the server with a status code, possibly followed by a Web resource.

Number	Meaning
200	OK
301-303	Page has moved (temporarily or permanently)
403	It is forbidden to access this page
404	Page not found
500	Internal server error

For secure communication across the Internet, the Secure HTTP (HTTPS) protocol is used for accessing or posting web server information. HTTPS can use authentication and encryption to secure data as it travels between the client and server. HTTPS specifies additional rules for passing data between the Application Layer and the Transport Layer.

When contacting an HTTP server to download a web page, a uniform resource locator (URL) is used to locate the server and a specific resource. The URL identifies:

- Protocol being used
- Domain name of the server needing to be accessed
- Location of the resource on the server, such as

  http://example.com/example1/index.htm

Many web server applications are available that allow for short URLs. Short URLs are popular because they are easier to write down, remember, or share. With a short URL, a default resource page is assumed when a specific URL is typed. When a user types in a shortened URL, like http://example.com, the default page that is sent to the user is actually the http://example.com/example1/index.htm web page.

To safeguard data, especially confidential information, some ISPs provide secure web services. To support secure web services ISPs, use HTTPS (HTTP over secure sockets layer (SSL)). HTTPS uses the same client request-server response process as HTTP, but the data stream is encrypted with SSL before being transported across the network.

When the HTTP data stream arrives at the server, the TCP layer passes it up to SSL in the server's Application Layer, where it is decrypted.

The maximum number of simultaneous connections that a server can support for HTTPS is less than that for HTTP. HTTPS creates additional load and processing time on the server due to the encryption and decryption of traffic. To keep server performance up, HTTPS should only be used when necessary, such as when exchanging confidential information.

FTP implementations include the functions of a protocol interpreter (PI) and a data transfer process (DTP)

File Transfer Protocol (FTP)

FTP is a connection-oriented protocol that uses TCP to communicate between a client FTP process and an FTP process on a server. FTP implementations include the functions of a protocol interpreter (PI) and a data transfer process (DTP). PI and DTP define two separate processes that work together to transfer files. As a result, FTP requires two connections to exist between the client and server, one to send control information and commands, and a second one for the actual file data transfer.

Protocol Interpreter (PI)

The PI function is the main control connection between the FTP client and the FTP server. It establishes the TCP connection and passes control information to the server. Control information includes things such as commands to navigate through a file hierarchy, as well as renaming or moving files. The control connection, or control stream, stays open until closed by the user. When a user wants to connect to an FTP server:

1. The user-PI sends a connection request to the server-PI on well-known port 21.

2. The server-PI replies and the connection is established.

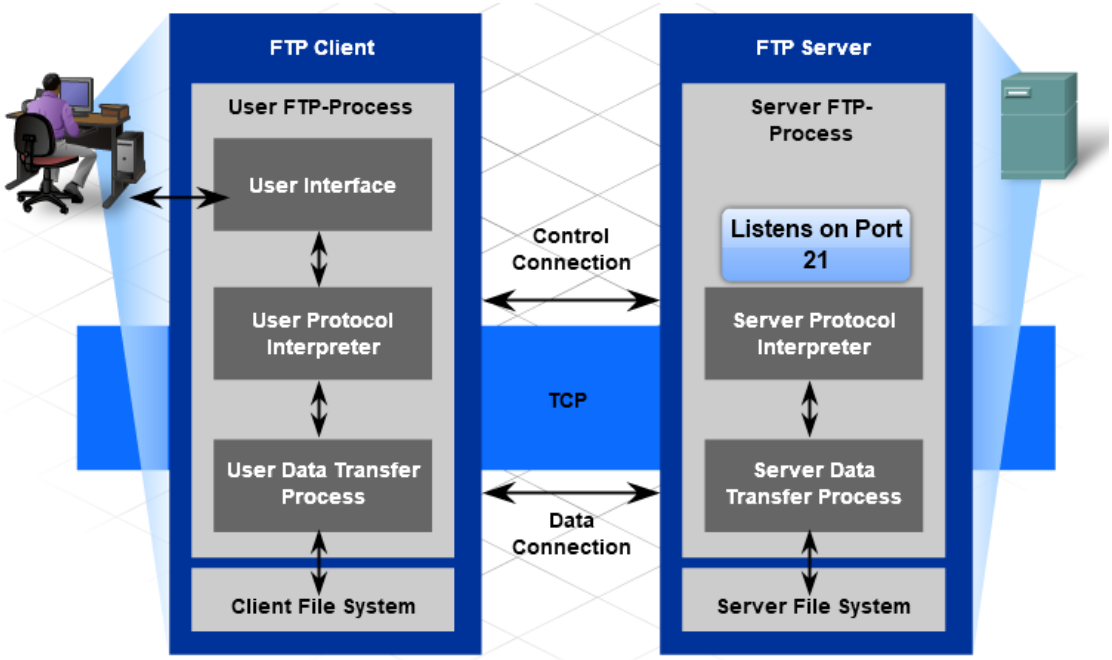
3. With the TCP control connection open, the server-PI process begins the login sequence.

4. The user enters credentials through the user interface and completes authentication.

5. Now the data transfer process can begin.

Data Transfer Process (DTP)

DTP is a separate data transfer function. This function is enabled only when the user wants to actually transfer files to or from the FTP server. Unlike the PI connection, which remains open, the DTP connection closes automatically when the file transfer is complete.



Email Protocols

One of the primary services offered by an ISP is email hosting. Email is a store and forward method of sending, storing, and retrieving electronic messages across a network. Email

messages are stored in databases on mail servers. ISPs often maintain mail servers that support many different customer accounts.

Email clients communicate with mail servers in order to send and receive email. Mail servers communicate with other mail servers to transport messages from one domain to another. In other words, an email client does not communicate directly with another email client when sending email. Both clients must rely upon the mail server for transport of the messages. This is true even when both users are in the same domain.

Email clients send messages to the email server configured in the application settings. When the server receives the message, it checks to see if the recipient domain is located on its local database. If it is not, it sends a DNS request to determine the mail server for the destination domain. Once the IP address of the destination mail server is known, the email is sent to the appropriate server.

Email supports three separate protocols for operation: SMTP, POP3, and IMAP4. The Application Layer process that sends mail, either from a client to a server or between servers, implements the SMTP protocol. A client retrieves email using one of two application layer protocols: POP3 or IMAP.

Simple Mail Transfer Protocol (SMTP)

The functions specified by the Simple Mail Transfer Protocol (SMTP) enable the transfer of mail reliably and efficiently. For SMTP applications to do this, two conditions must be met:

The mail message must be formatted properly

SMTP processes must be running on both client and server

SMTP message formats require a message header and a message body. While the message body can contain any amount of text, the message header must have a properly formatted recipient email address and a sender address. Any other header information is optional.

When a client sends email, the client SMTP process connects with a server SMTP process on well-known port 25. Once the connection is made, the client attempts to send mail to the server across the connection. Once the server receives the message, it either places the message in a

local account or forwards the message using the same SMTP connection process to another mail server.

The destination email server may not be online, or may be busy, when email messages are sent. Therefore, SMTP provides for the spooling of messages to be sent at a later time. Periodically, the server checks the queue for messages and attempts to send them again. After a predetermined expiration time, if the message is still undelivered, it will be returned to the sender as undeliverable.

One of the required fields in an email message header is the recipient email address. The structure of an email address includes the email account name or an alias, as well as the domain name of the mail server. An example of an email address:

recipient@gmail.com.

The @ symbol separates the account and the domain name of the server.

When a message is sent to recipient@gmail.com, the domain name is sent to the DNS server in order to obtain the IP address of the domain mail server. Mail servers are identified in DNS by an MX record indicator. When the destination mail server receives the message, it stores the message in the appropriate mailbox. The mailbox location is determined based on the account specified in the first part of the email address, in this case, the recipient account. The message will remain in the mailbox until the recipient connects to the server to retrieve the email.

If the mail server receives an email message that references an account that does not exist, the email is returned to the sender as undeliverable.

Post Office Protocol (POP)

The Post Office Protocol - Version 3 (POP3) is used to enable a workstation to retrieve mail from a mail server. With POP3, mail is downloaded from the server to the client and then deleted on the server.

The server starts the POP3 service by passively listening on TCP port 110 for client connection requests. When a client wishes to make use of the service, it sends a request to establish a TCP connection with the server. Once the connection is established, the POP3 server sends a

greeting. The client and POP3 server then exchange commands and responses until the connection is closed or aborted.

Since email messages are downloaded to the client and removed from the server, this means that there is not a centralized location where email messages are kept. This makes the POP3 protocol undesirable in a centralized backup solution for a small business.

The POP3 protocol is desirable for an ISP since it alleviates the ISP's responsibility of managing large amounts of storage for their email servers.

Internet Message Access Protocol (IMAP)

Internet Message Access Protocol (IMAP4) is another protocol that describes a method to retrieve email messages. However, unlike POP3, when the user connects to an IMAP-capable server, copies of the messages are downloaded to the client application. The original messages are kept on the server until manually deleted. Users view copies of the messages in their email client software.

1.3. Web Design

Web design is the process of creating websites which is front end design. It encompasses several different aspects, including webpage layout, content production, and graphic design using the different technology. Web designers build webpages using HTML tags that define the content and metadata of each page. Most websites include a combination of HTML and CSS(Cascading Style Sheet) that defines how each page will appear in a browser. Designer must keep in mind that website run on all platform and browser.

Website design means planning, creation and updating of websites. Website design also involves information architecture, website structure, user interface, navigation, website layout, colors, contrasts, fonts and imagery (photography) as well as icons design.

What is web Development?

Web development usually refers to the main non-design aspects of building web sites like writing markup and coding. Basically, Web development ranges from creating plain text pages to complex Web-based applications, social network applications and electronic business applications.

You can understand it by following hierarchy.

- Client-side coding
- Server-side coding
- Database technology

There are two broad divisions of web development – front-end development (also called client-side development) and back-end development (also called server-side development).

Front-end development refers to constructing what a user sees when they load a web application – the content, design and how you interact with it. This is done with three codes – HTML, CSS and JavaScript.

You can use following technology or platform to develop a website.

- PHP, Ruby on Rails, Asp.net, Perl, Java, Node.js, Python etc

Web Designer and Developer

Web designers are those who transform an idea or a story into a visually appealing design and use their layout to build the user experience throughout the whole website. They *design* the website's look and feel. As an architect would create a plan of your house prior to start building it, similarly a web designer would model the layout of your website before a web developer can start *developing* it.

Web developers sometimes called programmers, take the design created and build a fully functioning website. To put it (very) simply, think of the design as a non-interactive "picture" of a website. Developers take that design and break it up into it's components. They then either use just HTML or a more dynamic approach incorporating programming languages such as PHP to develop the various website pages. More advanced web developers may choose to utilize a Content Management System (CMS) like WordPress or Joomla in order to streamline development and allow clients an easy way to maintain and update their website.

Web developers may convert a static layout into a dynamic website by using image and content sliders, active states for links and buttons, and other interactive elements.

1.3.1. Website Design and Development Process

Despite conventional wisdom, the core part of website development and design is not necessary the coding process. Indeed, such technologies as HTML, CSS, and JavaScript give the web we know its shape and define the way we interact with the information. But what usually stay

behind the scenes and, at the same time, remain the crucial part of website development life cycle are the stages of preliminary information gathering, detailed planning, and post-launch maintenance.

There are numerous steps in the web site design and development process. From gathering initial information, to the creation of your web site, and finally to maintenance to keep your web site up to date and current. The exact process will vary slightly from designer to designer, but the basics are the same.

1. Information Gathering
2. Planning
3. Design
4. Development
5. Testing and Delivery
6. Maintenance

Phase 1: Information Gathering

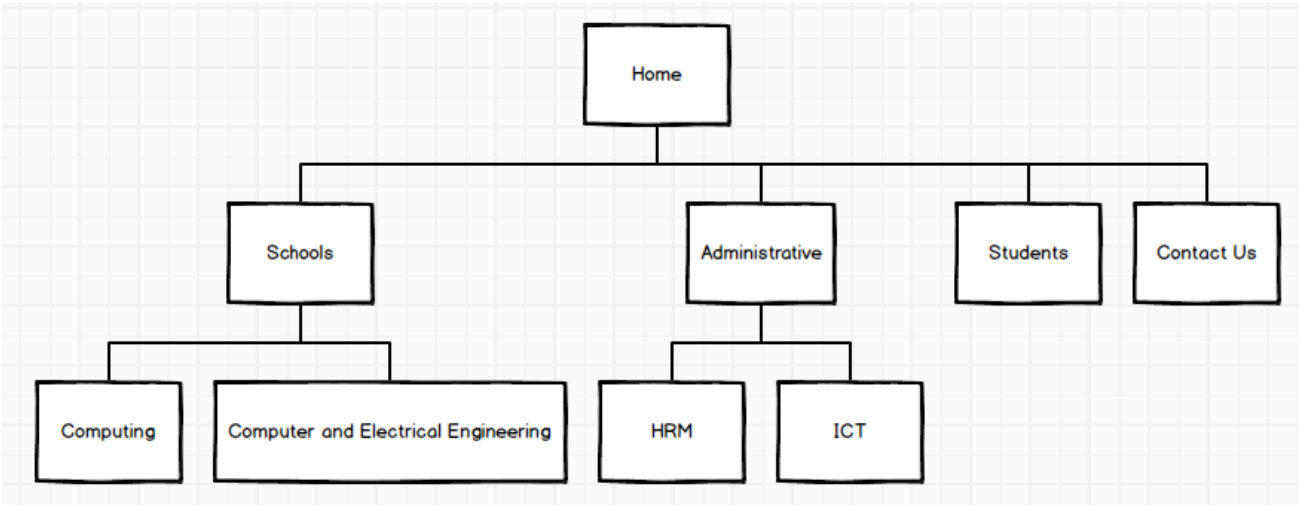
The first step in designing a successful web site is to gather information. Many things need to be taken into consideration when the look and feel of your site is created. This step is actually the most important one, as it involves a solid understanding of the company it is created for. **It involves a good understanding of *you*** – what your business goals and dreams are, and how the web can be utilized to help you achieve those goals. It is important that your web designer start off by asking a lot of questions to help them understand your business and your needs in a web site. Certain things to consider are:

- ✓ **Purpose:** What is the purpose of the site? Do you want to provide information, promote a service, sell a product … ?
- ✓ **Goals:** What do you hope to accomplish by building this web site? Two of the more common goals are either to make money or share information.
- ✓ **Target Audience:** Is there a specific group of people that will help you reach your goals? It is helpful to picture the "ideal" person you want to visit your web site. Consider their age, sex or interests – this will later help determine the best design style for your site.
- ✓ **Content:** What kind of information will the target audience be looking for on your site? Are they looking for specific information, a particular product or service, online ordering…?

Phase 2: Planning

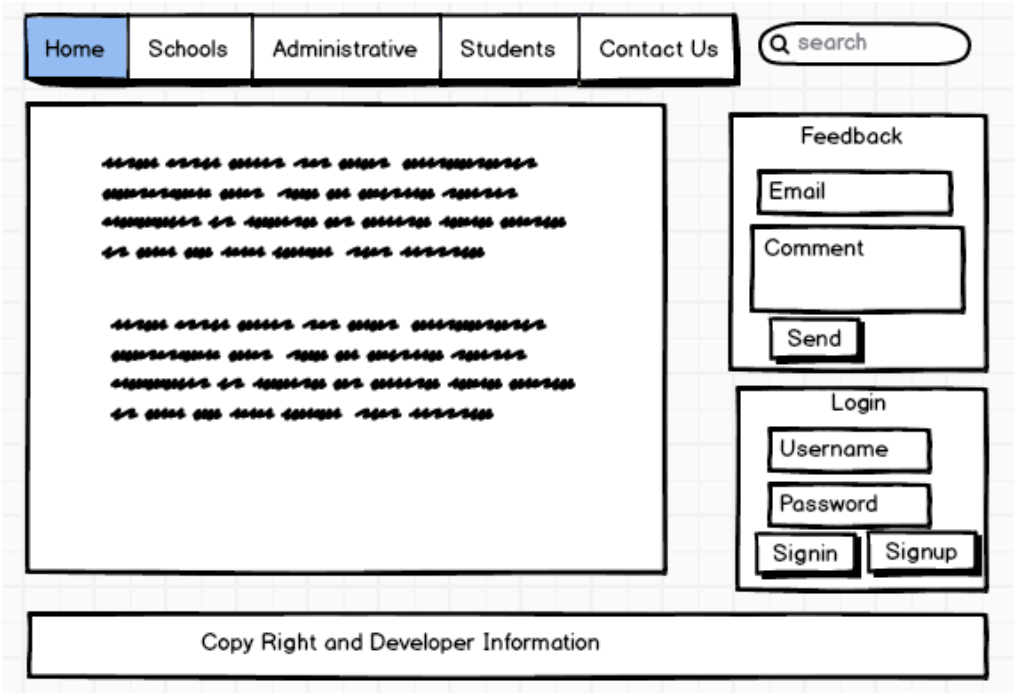
At this stage of website development cycle, the developer creates the data that can give to a customer an opportunity to judge how the entire site will look like.

On the basis of the information that was gathered together in the previous phase, the **sitemap** is created. The sitemap should describe the relations between the main areas of your website. Such representation could help understand how usable the final product will be. It can show you the "relationship" between the different pages of a website, so you can judge how easy it will be for the end-user to find the required information or service if he starts from the main page. The main reason behind the sitemap creation is to build a user-friendly and easy to navigate website.



The sitemap allows you to understand how the inner structure of a website looks like, but doesn't describe the user interface. Sometimes, before you start to code or even work on a design, there's a necessity to get approval from a customer that everything looks fine so you can begin the next phase of developing. In this case, a **wireframe** or **mock-up** is created. A wireframe is a visual representation of user interface that you're going to create. But it doesn't

contain any design elements such as colors, logos, etc. It only describes the elements that will be added to the page and their location. It's artless and cheap in production sketch.



The other important thing is select technology stack – programming language, frameworks, CMS that you're going to use.

Phase 3: Design

Drawing from the information gathered up to this point, it's time to determine the look and feel of your site. During the design phase, your website takes shape. All the visual content, such as images, photos, and videos is created at this step. Once again, all the info that was gathered through the first phase is crucial. The customer and target audience must be kept in mind while you work on a design.

Website layout is the result of designer's work. It can be a graphic sketch or an actual graphic design. The primary function of the layout is to represent the information structure, visualize the content, and demonstrate the basic functional. Layouts contain colors, logos, images and can give a general understanding of the future product.

After that, the customer can review the layout and send you his feedback. If the client is not sure about some aspects of your design, you should change the layout and send it back to him. This cycle should be repeated until the customer is completely satisfied.

Phase 4: Development

At this step, you can finally start creating the website itself. Graphic elements that have been designed during the previous stages should be used to create an actual website. Usually, the home page is created first, and then all sub-pages are added, according to the website hierarchy that was previously created in the form of a sitemap. Frameworks and CMS should be implemented to make sure that server can handle the installation and set-up smoothly.

All static web page elements that were designed during the mock-up and layout creation should be created and tested. Then, special features and interactivity should be added. A deep understanding of every website development technology that you're going to use is crucial at this phase.

Phase 5: Testing and Delivery

Testing is probably the most routine part of a process. You should check every form, every script, run a spell-checking software to find possible errors. Use code validations to check if your code follows the current web standards. Valid code is necessary, for example, if cross-browser compatibility is important for you.

After you check and re-check your website, it's time to upload it to a server. An FTP (File Transfer Protocol) software is used for that purpose. After you deployed the files, you should run yet another, final test to be sure that all your files have been installed correctly.

Phase 6: Maintenance

What's important to remember is that a website is more a service than a product. It's not enough to "deliver" a website to a user. You should also make sure that everything works fine, and everybody is satisfied and always be prepared to make changes in another case.

Feedback system added to the site will allow you to detect possible problems the end-users face. The highest priority task in this case is to fix the problem as fast as you can. If you won't, you may find one day that your users prefer to use another website rather than put up with the inconvenience. The other important thing is keeping your website up to date.

1.3.2. Web Design and Development Principles

The most important thing to keep in mind is that design is about communication. If you create a web site that works and presents information well, but looks ugly or doesn't fit with the client's brand, no one will want to use it. Similarly, if you make a beautiful web site that isn't usable and accessible, people may not be able to use it. Indeed, the elements and functionality of a finished web site design should work as a single cohesive unit.

Here is the list of 8-good design principles which will make your website aesthetic, user-friendly, effective, and engaging:

1. Simple is the best

Over-designed website may not work. Putting too many elements on the page may lead to distract visitors from the main purpose of your website. Simplicity always works in an effective web page design. Clean and fresh design of your website not only makes the website appealing, but also help user to navigate from one page to another seamlessly. Loading a website having design features which do not serve the purpose may be frustrating. Keep your design as simple as possible so that the visitors can feel it easy-to-use and can find their ways easily.

2. Consistency

Consistency in website design matter a lot. Give your attention to match design elements throughout each of the pages. It can be understood that your fonts, sizes, headings, sub-headings, and button styles must be the same throughout the website. Plan everything in advance. Finalize the fonts and the right colors for your texts, buttons etc, and stick to them throughout the development. CSS (Cascading Style Sheets) would come in handy to keep the complete information about design styles and elements.

3. **Typography & Readability**

No matter how good your design is text still rules the website as it provides users the desired information. You should keep your typography visually appealing and readable for visitors, Consider using fonts that are easier to read. The modern sans-serif fonts as Ariel, Helvetica etc. can be used for the body texts. Make proper combinations of typefaces for each and every design element such as headlines, body texts, buttons etc.

4. Mobile compatibility

Keeping in mind the ever-growing usage of smartphones, tablets and phablets, web design must be effective for various screens. If your website design doesn't support all screen sizes, chance is that you'll lose the battle to your competitors. There are a number of web design studios or service points from where you can turn your desktop design into a responsive and adaptive one for all screen sizes.

5. Color palette and imagery

A perfect color combination attracts users while a poor combination can lead to distraction. This necessitates you to pick a perfect color palette for your website which can create a pleasing atmosphere, thus leaving a good impact on visitors. Enhance users experience by selecting complementary color palette to give a balanced-look to your website design. Remember to white space use as it avoids your website from visual clutter and mess. Also avoid using too many colors. 3 or 4 tones for the whole websites are ample to give appealing and clear design. Same is the case with images. Don't use multiple vibrant images

6. Easy loading

No one likes the website that takes too much time to load. So take care of it by optimizing image sizes, combing code into a central CSS or JavaScript file as it reduces HTTP requests. Also, compress HTML, JavaScript and CSS for enhanced loading speed.

7. Easy Navigation

Study shows that visitors stay more time on the websites having easy navigation. For effective navigation, you may consider creating a logical page hierarchy, using bread scrums, and designing clickable buttons. You should follow the "three-click-rule" so that visitors can get the required information within three clicks.

8. Communication

The ultimate purpose of the visitors is to get information, and if your website is able to communicate your visitors efficiently, most probably they would spend more time on your website. Tricks that may work to establish effortless communication with the visitors are – organizing information by making good use of headlines and sub-headlines, cutting the waffle, and using bullet points, rather than long gusty sentences.

Keeping the above principles, you can easily develop an aesthetic and functional website. Without this base, it would be difficult to travel a long path. Only with a clean and user-friendly design, you can think to succeed.

Chapter One
Introduction to Internet and the Web

1.1. Internet history, Uses and Services

What is Internet?

The Internet is a worldwide system of interconnected computer networks. The computers and computer networks exchange information using TCP/IP (Transmission Control Protocol/Internet Protocol) to communicate with each other. The computers are connected via the telecommunications networks, and the Internet can be used for e-mailing, transferring files and accessing information on the World Wide Web.

Internet History

The rapid development of Internet started at the early 1960, paralleled with the developments of the computers. Those scientists and researcher started to realize a great vision, a future that everyone will be able to communicate by using their computers.

J.C.R. Licklider of MIT, first proposed a global network of computers in 1962 and followed by Leonard Kleinrock from MIT, who published the first paper on packet switching theory.

ARPANET (Advanced Research Projects Agency NETwork), which is the former of Internet, was a project launched by Military Department of USA. It was brought online at Oct 29, 1969 by Charley Kline at UCLA, when he attempted to perform a remote login from UCLA to SRI. In order to get attentions from public, they made the first public demonstration of ARPANET at an international conference at October 1972.

The initial ARPANET was a single, closed network. In order to communicate with an ARPANET, one had to be attached to another ARPANET IMP (interface message processor). Hence, the disadvantages of single network were realized and lead the development of open-architecture network and also different protocols of internetworking, which enable multiple networks, can be joined together. E-mail was adapted for ARPANET by Ray Tomlinson of BBN in 1972. The telnet protocol, enabling logging on to a remote computer, was published as a Request for Comments (RFC) in 1972. RFC's are a means of sharing developmental work throughout community. The FTP protocol, enabling file transfers between Internet sites, was published as an RFC in 1973, and from then on RFC's were available electronically to anyone who had use of the FTP protocol.

Before the TCP/IP protocol was introduced by BoB Kahn, the networking protocols used for the ARPANET was NCP, Network Control Protocols. NCP did not have the ability to address networks further downstream than a destination IMP on ARPANET. By 1980, the Internet had reached a certain level of maturity and started to exposed to public usage more and more often. At the same time, French launched the Minitel project to bring data networking into everyone's home by gave away a free terminal to each household requested.

At the 1990s, the Internet predecessor, ARPANET finally came to an end, and replaced by the NSFNET (National Science Foundation) which serve as a backbone connecting regional networks in USA. However, the most significant changes of Internet at 1990s was the World Wide Web (WWW) application which truly brought Internet to our daily life. Various technologies such as VoIP, HTML, web browsers with graphical user interfaces, P2P file sharing, and instant messaging which is very familiar to us nowadays.

Advantages

Internet covers almost every aspect of life, one can think of. Here, we will discuss some of the advantages of Internet:

- Social Networking
- Education and Technology
- Entertainment
- Online Services

Internet allows us to communicate with the people sitting at remote locations. There are various apps available on the wed that uses Internet as a medium for communication. One can find various social networking sites such as: Facebook, Twitter, Yahoo, Google+, Flickr, Orkut etc

One can surf for any kind of information over the internet. Information regarding various topics such as Technology, Health & Science, Social Studies, Geographical Information, Information Technology, Products etc can be surfed with help of a search engine.

Apart from communication and source of information, internet also serves a medium for entertainment. Following are the various modes for entertainment over internet. These are Online Television, Online Games, Songs, Videos and Social Networking Apps.

Internet allows us to use many services like Internet Banking, Matrimonial Services, Online Shopping, Online Ticket Booking, Online Bill Payment, Data Sharing & E-mail.

Internet provides concept of **electronic commerce** that allows the business deals to be conducted on electronic systems

Disadvantages

However, Internet has proved to be a powerful source of information in almost every field, yet there exists many disadvantages discussed below:

- **Threat to Personal Information**: There are always chances to lose personal information such as name, address, credit card number. Therefore, one should be very careful while sharing such information. One should use credit cards only through authenticated sites.
- **Spamming:** it corresponds to the unwanted e-mails in bulk. These e-mails serve no purpose and lead to obstruction of entire system.
- **Virus attack**: Viruses can easily be spread to the computers connected to internet. Such virus attacks may cause your system to crash or your important data may get deleted.
- Also a biggest threat on internet is pornography. There are many pornographic (sexual sit) sites that can be found, letting your children to use internet which indirectly affects the children healthy mental life.
- **Cyber Crime**

1.1.1. The World Wide Web (WWW)

WWW is also known as W3. It offers a way to access documents spread over the several servers over the internet. These documents may contain texts, graphics, audio, video, hyperlinks. The hyperlinks allow the users to navigate between the documents.

The World Wide Web is a system of **Internet servers** that use HTTP (Hypertext Transfer Protocol) to transfer documents formatted in HTML (Hypertext Mark-up Language). These are viewed by using software for web browsers such as Netscape, Safari, Google Chrome and Internet Explorer. Hypertext enables a document to be connected to other documents on the web through hyperlinks. It is possible to move from one document to another by using hyperlinked text found within web pages.

1.1.2. Browser and Web Server

Web Browser is application software that allows us to view and explore information on the web. User can request for any web page by just entering a URL into address bar.

Web browser can show text, audio, video, animation and more. It is the responsibility of a web browser to interpret text and commands contained in the web page.

Earlier the web browsers were text-based while now days graphical-based or voice-based web browsers are also available. Following are the most common web browser available today:

Browser	**Vendor**
Internet Explorer	Microsoft
Google Chrome	Google
Mozilla Firefox	Mozilla
Netscape Navigator	Netscape Communications Corp.
Opera	Opera Software
Safari	Apple
Sea Monkey	Mozilla Foundation
K-meleon	K-meleon

Web Server

Web server is a computer where the web content is stored. Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.

Web site

is collection of web pages while web server is software that responds to the request for web resources.

Web Server Working

Web server respond to the client request in either of the following two ways:

- Sending the file to the client associated with the requested URL.
- Generating response by invoking a script and communicating with database

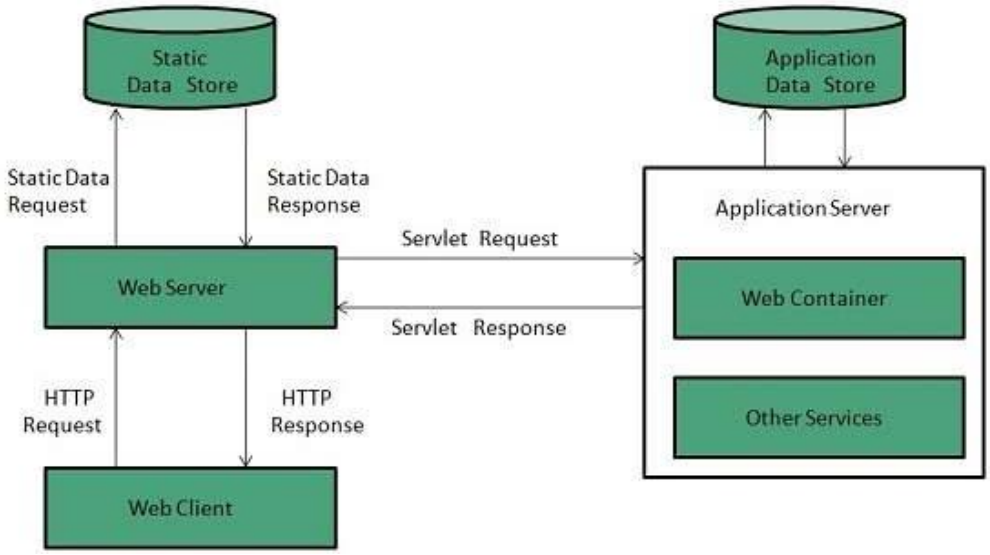


Figure 1.1.2 Client Server Architecture

1.1.3. Document Types and MIME

A **media type** (formerly known as **MIME type**) is a two-part identifier for file formats (formatted file) and format contents transmitted on the Internet. The Internet Assigned Numbers Authority (IANA) is the official authority for the standardization and publication of these classifications. Media types were originally defined in Request for Comments 2045 in November 1996 as a part of *MIME (Multipurpose Internet Mail Extensions)* specification, for denoting type of email message content and attachments; hence the name *MIME type*. Media types are also used by other internet protocols such as HTTP and document file formats such as HTML, for similar purpose.

A media type consists of a *type* and a *subtype*, which is further structured into a *tree*. A media type can optionally define a *suffix* and *parameters*:

```
type "/" [tree "."] subtype ["+" suffix] *[";" parameter]
```

The currently registered types are: `application`, `audio`, `example`, `font`, `image`, `message`, `model`, `multipart`, `text` and `video`.

As an example, an HTML file might be designated `text/html; charset=UTF-8`. In this example, `text` is the type, `html` is the subtype, and `charset=UTF-8` is an optional parameter indicating the character encoding.

A subtype typically consists of a media format, but it may or must also contain other content, such as a tree prefix, producer, product or suffix, according to the different rules in registration trees.

Types, subtypes, and parameter names are case-insensitive. Parameter values are usually case-sensitive, but may be interpreted in a case-insensitive fashion depending on the intended use.

Common examples

- `application/javascript`
- `application/xml`
- `application/zip`
- `application/pdf`
- `application/sql`
- `application/msword` (.doc)
- `application/vnd.openxmlformats-officedocument.wordprocessingml.document`(.docx)
- `application/vnd.ms-excel` (.xls)
- `application/vnd.openxmlformats-officedocument.spreadsheetml.sheet` (.xlsx)
- `application/vnd.ms-powerpoint` (.ppt)
- `application/vnd.openxmlformats-officedocument.presentationml.presentation` (.pptx)
- `application/vnd.oasis.opendocument.text` (.odt)
- `audio/mpeg`
- `audio/ogg`
- `multipart/form-data`
- `text/css`
- `text/html`
- `text/xml`
- `text/csv`
- `text/plain`
- `image/png`
- `image/jpeg`
- `image/gif`

1.1.4. Uniform Resource Locators (URLs)

Uniform Resource Locator (URL) refers to a web address which uniquely identifies a document over the internet. This document can be a web page, image, audio, video or anything else present on the web. For example: www.dmu.edu.et**/index.html**.

URL Types

There are two forms of URL as listed:

- Absolute URL
- Relative URL

Absolute URL

Absolute URL is a complete address of a resource on the web. This completed address comprises of protocol used, server name, path name and file name.

For example http:// www.dmu.edu.et/index.htm. where:

- **http** is the protocol.
- **dmu.edu.et** is the server name(domain name).
- **index.htm** is the file name.

The protocol part tells the web browser how to handle the file. Similarly we have some other protocols also that can be used to create URL are: FTP, https, Gopher, mailto, news

Relative URL

Relative URL is a partial address of a webpage. Unlike absolute URL, the protocol and server part are omitted from relative URL. **Relative URLs are used for internal links** i.e. to create links to file that are part of same website as the WebPages on which you are placing the link. For example, to link an image on dmu.edu.et/bids/bids.pdf, we can use the relative URL which can take the form like **/ bids/bids.pdf**.

Difference between Absolute and Relative URL

Absolute URL	Relative URL
• Used to link web pages on different websites • Difficult to manage. • Changes happen when the server name or directory name changes • Take time to access	• Used to link web pages within the same website. • Easy to Manage • Remains same even of we change the server name or directory name. • Comparatively faster to access.

1.1.5. Domain Name System (DNS)

DNS is an application layer protocol used to resolve hostnames to IP addresses. Although a host can be accessed by using only its IP address, DNS makes your life easier by using domain names. For example, you can access the Google website by typing http://208.117.229.214 in your browser, but it is much easier to type http://www.google.com.

Each host that wants to use DNS needs to have a DNS server configured. When you type a URL in your browser (e.g. http://www.google.com) , your host will query the DNS server for the IP address of www.google.com. The DNs server will resolve the query and send the answer back to the host. The host will then be able to establish a connection to http://www.google.com.

1.2. Communication Protocols

In telecommunication, a **communication protocol** is a system of rules that allow two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both.

Communicating systems use well-defined formats for exchanging various messages. Each message has an exact meaning intended to elicit a response from a range of possible responses pre-determined for that particular situation. The specified behavior is typically independent of how it is to be implemented. Communication protocols have to be agreed upon by the parties involved. To reach agreement, a protocol may be developed into a technical standard. A programming language describes the same for computations, so there is a close analogy between protocols and programming languages: *protocols are to communication what programming languages are to computations.*

Multiple protocols often describe different aspects of a single communication. A group of protocols designed to work together are known as a protocol suite; when implemented in software they are a protocol stack.

1.2.1. TCP/ IP, HTTP, FTP and Email Protocols

TCP

TCP is a reliable, guaranteed-delivery protocol. TCP specifies the methods hosts use to acknowledge the receipt of packets, and requires the source host to resend packets that are not acknowledged. TCP protocols also govern the exchange of messages between the source and

destination hosts to create a communication session. TCP is often compared to a pipeline, or a persistent connection, between hosts. Because of this, TCP is referred to as a connection-oriented protocol.

TCP requires overhead to keep track of the individual conversations between source and destination hosts and to process acknowledgements and retransmissions. In some cases, the delays caused by this overhead cannot be tolerated by the application. These applications are better suited to UDP.

UDP

UDP is a very simple, connectionless protocol. It has the advantage of providing for low overhead data delivery. Because UDP is a "best effort" Transport Layer protocol, UDP datagrams may arrive at the destination out of order, or may even be lost all together. UDP does not provide guaranteed data delivery or flow control. Applications that use UDP can tolerate small amounts of missing data. An example of a UDP application is Internet radio. If a piece of data is not delivered, there may only be a minor effect on the quality of the broadcast.

TCP Vs UDP

UDP is a very simple protocol. Because it is not connection-oriented and does not provide the sophisticated retransmission, sequencing, and flow control mechanisms of TCP, UDP has a much lower overhead.

UDP is often referred to as an unreliable delivery protocol; because there is no guarantee that a message has been received by the destination host. This does not mean that applications that use UDP are unreliable. It simply means that these functions are not provided by the Transport Layer protocol and must be implemented elsewhere if required.

Although the total amount of UDP traffic found on a typical network is often relatively low, key Application Layer protocols that use UDP include:

- Domain Name System (DNS)
- Simple Network Management Protocol (SNMP)
- Dynamic Host Configuration Protocol (DHCP)
- Routing Information Protocol (RIP)
- Trivial File Transfer Protocol (TFTP)
- Online games

HTTP and HTTPs

The Hypertext Transfer Protocol (HTTP), one of the protocols in the TCP/IP suite, was originally developed to enable the retrieval of HTML formatted web pages. It is now used for distributed, collaborative information sharing. The HTTP protocol has evolved through multiple versions. The version currently used by most ISPs to provide web-hosting services is HTTP version 1.1. Unlike earlier versions, this version enables a single web server to host multiple web sites. It also permits persistent connections, so that multiple request and response messages can use the same connection, reducing the time it takes to initiate new TCP sessions.

HTTP specifies a request/response protocol. When a client, typically a web browser, sends a request message to a server, the HTTP protocol defines the message types the client uses to request the web page. The HTTP protocol also defines the message types the server uses to respond.

Although it is remarkably flexible, HTTP is not a secure protocol. The request messages send information to the server in plain text that can be intercepted and read. Similarly, the server responses, typically HTML pages, are also sent unencrypted.

Some HTTP cmmands (your browser sends these internally)

- GET resource -- requests data from a specified resource
- POST resource -- submits data to be processed to a specified resource
- PUT resource -- uploads a representation of the specified URL
- DELETE resource -- deletes the specified resource `

HTTP Status Codes

When a request is made by the browser, a response is sent back by the server with a status code, possibly followed by a Web resource.

Number	Meaning
200	OK
301-303	Page has moved (temporarily or permanently)
403	It is forbidden to access this page
404	Page not found
500	Internal server error

For secure communication across the Internet, the Secure HTTP (HTTPS) protocol is used for accessing or posting web server information. HTTPS can use authentication and encryption to

secure data as it travels between the client and server. HTTPS specifies additional rules for passing data between the Application Layer and the Transport Layer.

When contacting an HTTP server to download a web page, a uniform resource locator (URL) is used to locate the server and a specific resource.

The URL identifies:

- Protocol being used
- Domain name of the server needing to be accessed
- Location of the resource on the server, such as http://example.com/example1/index.htm

Many web server applications are available that allow for short URLs. Short URLs are popular because they are easier to write down, remember, or share. With a short URL, a default resource page is assumed when a specific URL is typed. When a user types in a shortened URL, like http://example.com, the default page that is sent to the user is actually the http://example.com/example1/index.htm web page.

To safeguard data, especially confidential information, some ISPs provide secure web services. To support secure web services ISPs use HTTPS (HTTP over secure sockets layer (SSL)). HTTPS uses the same client request-server response process as HTTP, but the data stream is encrypted with SSL before being transported across the network.

When the HTTP data stream arrives at the server, the TCP layer passes it up to SSL in the server's Application Layer, where it is decrypted.

The maximum number of simultaneous connections that a server can support for HTTPS is less than that for HTTP. HTTPS creates additional load and processing time on the server due to the encryption and decryption of traffic. To keep server performance up, HTTPS should only be used when necessary, such as when exchanging confidential information.

File Transfer Protocol (FTP)

FTP is a connection-oriented protocol that uses TCP to communicate between a client FTP process and an FTP process on a server. FTP implementations include the functions of a protocol interpreter (PI) and a data transfer process (DTP). PI and DTP define two separate processes that work together to transfer files. As a result, FTP requires two connections to exist between the client and server, one to send control information and commands, and a second one for the actual file data transfer.

Protocol Interpreter (PI)

The PI function is the main control connection between the FTP client and the FTP server. It establishes the TCP connection and passes control information to the server. Control information includes things such as commands to navigate through a file hierarchy, as well as renaming or moving files. The control connection, or control stream, stays open until closed by the user. When a user wants to connect to an FTP server:

1. The user-PI sends a connection request to the server-PI on well-known port 21.

2. The server-PI replies and the connection is established.

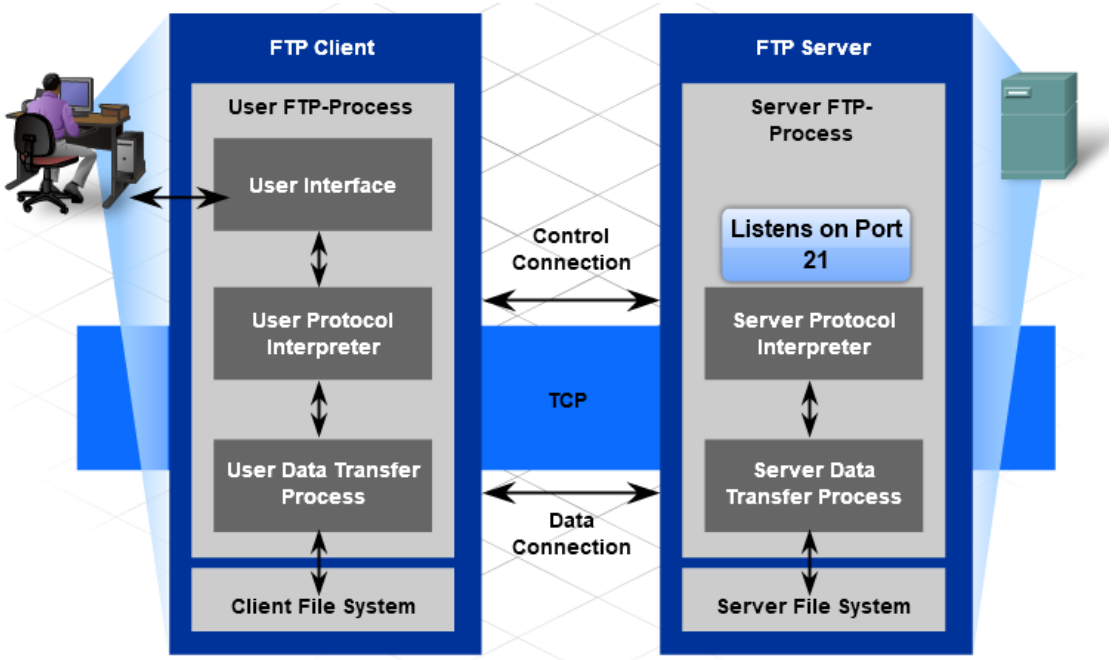
3. With the TCP control connection open, the server-PI process begins the login sequence.

4. The user enters credentials through the user interface and completes authentication.

5. Now the data transfer process can begin.

Data Transfer Process (DTP)

DTP is a separate data transfer function. This function is enabled only when the user wants to actually transfer files to or from the FTP server. Unlike the PI connection, which remains open, the DTP connection closes automatically when the file transfer is complete.



Email Protocols

One of the primary services offered by an ISP(internet service provider) is email hosting. Email is a store and forward method of sending, storing, and retrieving electronic messages across a

network. Email messages are stored in databases on mail servers. ISPs often maintain mail servers that support many different customer accounts.

Email clients communicate with mail servers in order to send and receive email. Mail servers communicate with other mail servers to transport messages from one domain to another. In other words, an email client does not communicate directly with another email client when sending email. Both clients must rely upon the mail server for transport of the messages. This is true even when both users are in the same domain.

Email clients send messages to the email server configured in the application settings. When the server receives the message, it checks to see if the recipient domain is located on its local database. If it is not, it sends a DNS request to determine the mail server for the destination domain. Once the IP address of the destination mail server is known, the email is sent to the appropriate server.

Email supports three separate protocols for operation: SMTP, POP3, and IMAP4. The Application Layer process that sends mail, either from a client to a server or between servers, implements the SMTP protocol. A client retrieves email using one of two application layer protocols: POP3 or IMAP.

Simple Mail Transfer Protocol (SMTP)

The functions specified by the Simple Mail Transfer Protocol (SMTP) enable the transfer of mail reliably and efficiently.

For SMTP applications to do this, two conditions must be met:

- ✓ The mail message must be formatted properly
- ✓ SMTP processes must be running on both client and server

SMTP message formats require a message header and a message body. While the message body can contain any amount of text, the message header must have a properly formatted recipient email address and a sender address. Any other header information is optional.

When a client sends email, the client SMTP process connects with a server SMTP process on well-known port 25. Once the connection is made, the client attempts to send mail to the server across the connection. Once the server receives the message, it either places the message in a local account or forwards the message using the same SMTP connection process to another mail server.

The destination email server may not be online, or may be busy, when email messages are sent. Therefore, SMTP provides for the spooling of messages to be sent at a later time. Periodically, the server checks the queue for messages and attempts to send them again. After a predetermined expiration time, if the message is still undelivered, it will be returned to the sender as undeliverable.

One of the required fields in an email message header is the recipient email address. The structure of an email address includes the email account name or an alias, as well as the domain name of the mail server. An example of an email address:

recipient@cisco.com.

The @ symbol separates the account and the domain name of the server.

When a message is sent to recipient@cisco.com, the domain name is sent to the DNS server in order to obtain the IP address of the domain mail server. Mail servers are identified in DNS by an MX record indicator. When the destination mail server receives the message, it stores the message in the appropriate mailbox. The mailbox location is determined based on the account specified in the first part of the email address, in this case, the recipient account. The message will remain in the mailbox until the recipient connects to the server to retrieve the email.

If the mail server receives an email message that references an account that does not exist, the email is returned to the sender as undeliverable.

Post Office Protocol (POP)

The Post Office Protocol - Version 3 (POP3) is used to enable a workstation to retrieve mail from a mail server. With POP3, mail is downloaded from the server to the client and then deleted on the server.

The server starts the POP3 service by passively listening on TCP port 110 for client connection requests. When a client wishes to make use of the service, it sends a request to establish a TCP connection with the server. Once the connection is established, the POP3 server sends a greeting. The client and POP3 server then exchange commands and responses until the connection is closed or aborted.

Since email messages are downloaded to the client and removed from the server, this means that there is not a centralized location where email messages are kept. This makes the POP3 protocol undesirable in a centralized backup solution for a small business.

The POP3 protocol is desirable for an ISP since it alleviates the ISP's responsibility of managing large amounts of storage for their email servers.

Internet Message Access Protocol (IMAP)

Internet Message Access Protocol (IMAP4) is another protocol that describes a method to retrieve email messages. However, unlike POP3, when the user connects to an IMAP-capable server, copies of the messages are downloaded to the client application. The original messages are kept on the server until manually deleted. Users view copies of the messages in their email client software.

Users can create a file hierarchy on the server to organize and store mail. That file structure is duplicated on the email client as well. When a user decides to delete a message, the server synchronizes that action and deletes the message from the server.

For small to medium-sized businesses, there are many advantages to the IMAP protocol. IMAP can lead to long-term storage of email messages on mail servers and allow for centralized backup. It also enables employees to access email messages from multiple locations, using different devices or client software. The mailbox folder structure that a user is used to seeing is available for viewing regardless of how the user accesses the mailbox.

For an ISP, IMAP may not be the protocol of choice. It can be expensive to purchase and maintain the disk space to support the large number of stored emails. Additionally, if customers expect their mailboxes to be backed up routinely, that can further increase the costs to the ISP.

Chapter Two

Introduction to html/ XHTM

2.1. HyperText Mark-up Language (HTML)

Internet programming includes broad variety of technologies spanning such diverse areas such as protocols for communication.

The word HTML, stands for **Hypertext Markup Language**, is the predominant markup language for web pages. A **markup language** is a set of markup tags, and HTML uses markup tags to describe web pages.

Mark-up means anything that is added to a document to add special meaning or provide extra information about it.

HTML is written in the form of HTML elements consisting of HTML **tags** surrounded by angle brackets (e.g. <html>) within the web page content.

HTML was invented in 1990 by a scientist called Tim Berners-Lee. An HTML file is a text file containing small **markup tags.** The markup tags tell the Web browser **how to display** the page.

Mark-up languages can be classified as one of these **four** types:

- **Stylistic** – indicating how the document should be presented e.g. adding bold or italics. HTML has tags e.g.<font>, <b> and <i> for stylistic mark-up - we'll see more of these when we start working with HTML.
- **Structural** – indicating how the document should be structured. In HTML, examples are <h1>, <h2> etc, <p> and .
- **Semantic** – informing about the content of the data. In HTML, <title> is an example of this.
- **Functional** – adds functionality to the marked-up data, e.g. hyperlinks in HTML, embedded sound files etc.

The main language used for writing web pages is HTML.

- HTML tells a browser what text and images to display and how to format the text e.g. show it in **bold** or *italics* or to show it in a particular colour.
- A HTML file is just a text file – so you can start writing HTML in a simple editor like Windows Notepad.
- The HTML language is defined as a set of **tags** that are used to surround the text that will appear on the web page or to precede a piece of text. For example, the tag that denotes the start of a web page is <html>.
- Most tags in HTML have a **start tag** and an **end tag**. The start tag looks like **<html>** and the end tag looks like **</html>**.

You can easily edit HTML files using a WYSIWYG (what you see is what you get) editor like FrontPage, Dreamweaver, Claris Home Page, or Adobe PageMill instead of writing your markup tags in a plain text file. But

- *If you want to design professional web pages using these tools will not be enough. You must be familiar with html codes.*
- *Results of these editors are big and sometimes chaotic code. Maintaining this code is very difficult.*
- *If you want to design dynamic web pages in future you will need to know html codes.*
- *If you need forms in your pages to send information to server and return result pages back to browser you will need to know html codes.*

The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

HTML tags usually come in pairs like and , but not always. The first tag in a pair is the start tag, the second tag is the end tag; which are refer for opening tags and closing tags. But, it doesn't mean all tags has both open and close tags because there are empty tags (self close tag) which has only open tag like <br> and <img> tags.

HTML elements form the building blocks of all websites, it allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can also embed scripts in languages such as JavaScript which affect the behavior of HTML webpage. HTML can also be used to include Cascading Style Sheets (CSS) to define the appearance and layout of text.

HTML tags are used to create the web page, and interpreted by any browser types such as internet explorer, nets cap, Mozilla, safari, etc. The purposes of these web browser types are to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

A very good quality associated with all the browsers is that they would not give any error if we have not put any HTML tag or attribute properly. They will just ignore the wrong tag or attribute and will apply only correct tags and attributes before displaying the result.

HTML document can be created using a simple text editor like **Notepad, WordPad, Notepade++, macromedia Dreamweaver Mx** or any other text editors.

For example: using Notepad, we can edit HTML program as follows:-:-

Step 1: Open Notepad.

Step 2: Writing the HTML tags in the new Notepad file, then save using a file extension of html

Step 3: Double-click on the HTML file to run

HTML is a language, which makes it possible to present information (e.g. scientific research) on the Internet.

Note that:- To see the HTML code of a page on the Internet, simply click "View" in the top menu of the browser and choose "Source, or right click on html file icon→open with→notepad

2.2. HTML Tags and Blocks/ skeleton of HTML codes

Container tags allow you to group and structure content on a webpage, while empty tags serve specific purposes like line breaks or inserting images. empty tag has no content

HTML Tags

- HTML elements are defined using HTML tags
- HTML tags are used to mark-up HTML elements
- HTML tags are surrounded by the two characters < and >, i.e. using angle brackets
- The text between the start and end tags is the element content
- There are two types of tags:-
  - Empty tag:- type of html tag which has not any close tags such as <br> and <img> tags

- Container tag: - tag types which have both open and close tags. For instance <body> and <html> tags.

N.B:- HTML tags are **not case sensitive;** <b> means the same as <B>. Both upper and lowercases of tags in html interpreted similarly. But, the latest web standards follow lowercase tags.

One of the uses of the META tag is to give search engines (like Google, Yahoo) information about the web page

Title tags are also important for our search engine point of view

Basic HTML codes

```
<! DOCTYPE html>
<HTML>
        <HEAD>
                <META NAME="keywords" CONTENT="Debre Markos, university,
                Ethiopia, DMU, education">
                <META NAME="description" CONTENT="Debre Markos University is
                one of the university located in Amhara region, Ethiopia">
                <TITLE>My Website – first page</TITLE>
        </HEAD>
        <BODY>
                <P>This is the body of the web page.</P>
        </BODY>
</HTML>
```

Descriptions of these tags

Tag	Description
<**DOCTYPE**>	This tag is called a ***prologue identifier***. It does not have to be on a web page and is often omitted. It indicates that the document uses the specified HTML version (in this case, version 5.0, which is the current version)
<**HTML**> </**HTML**>	This tag denotes the **beginning** and **end** of the web page
<**HEAD**> </**HEAD**>	This tag marks the header section of the page – it contains the page title and other information about the document
<**META**>	**Meta tags** must be contained within the **HEAD** tags and are used to provide information about the **page** that cannot be defined by any other tags. Such information can be extracted by web servers and clients for identifying, indexing and cataloguing documents. One of the uses of the META tag is to give **search**

	engines (like Google, Yahoo) information about the web page. The tags in the code above would mean that if someone searched for any of the 'keywords' (Debre Markos, university, Ethiopia, DMU, education) in a search engine, this page would be found. The search engine would show the text for the 'description' on the listing of found pages. Some search engines actively search the web for information – these will pick up the information from the **Meta tags**. Others allow you to submit information about your web site to them – for example, Yahoo provides a section to suggest a web site. There is also a web site called Submit I! (**www.submit-it.com**) that allows you to submit your web site to multiple search engines.
<**TITLE**> </**TITLE**>	The **title tag** is used to specify a title for the page. This title will display in the browser's title bar when the web page is displayed in a browser. The **title** should be consistent across all the pages in a web site – so that as a user can see in the **title bar** and the Windows **task bar** what site they are on. The title should also indicate what part of the site the page is in e.g. 'Debre Markos University – Departments'
<**BODY**> </**BODY**>	the body section is the part of the page that a user will see in a **browser** – it contains all the text, images etc that make up the page content, along with HTML tags to format the text. It can also be used to specify attributes that affect the whole page e.g. the background colour

HTML, HEAD, TITLE and BODY Tags are html main tags that are used to construct an html skeleton / structure. To develop the webpage, it is possible to use different webpage editors/ plain text editor or any advance editor. Whatever the use of the text editor the structure of basic HTML code/ tags is similar as shown above.

The total code is divided into two parts and both the parts are kept inside <html> tags. The first part is header part, which is optional, and the second one is body part which is the main part.

Any html code should start with <html> and should end with </html>. The header part should starts with <head> and ends with </head>, after this the body part starts with <body> and ends with </body> tag. Inside the body tag, we keep all our content which we want to display to webpage users. Whatever we place in this part of the page will be displayed by the browser to the web users.

The text or tags what goes inside the head tags will not be displayed in the browser window. Inside this tag we keep all the meta keyword tags used for search engines. One of the important tags is title tag and written as <title> title of the webpage</title> tag. Title tags are displayed at the top left side of the browser window. Title tags are also important for our search engine point of view. It is possible to keep most important keywords inside the title tag.

Html developers place web contents in between open and close body tags to display for visitors. So go on filling this place with content to show to the web user. Hence, the purpose of the <body> tag is to define the HTML element that contains the body of the HTML document.

The font of documents in between body tags could be changes using Font tags. It uses to change the way text looks by changing its font, size, colour and emphasis. Written as <font> </font>. It use different key words such as size, color, face, etc.

HTML common Elements and their Descriptions: - There are some Common html tags because some text may be more important than others.

Tag	Description
<h1> to <h6>	Defines header 1 to header 6
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a horizontal rule
<!-->	Defines a comment

- **HTML HEADING:-** HTML provides six headers, called header elements, for specifying the relative importance of information

– <h1> … </h1>, <h2> … </h2> to <h6> … </h6>

It is assumed the text in <h1> is more important than that in <h2>, <h2> is more important than <h3> and so on so forth based on the size of the text by taking the more important is bigger text. See the following example:-

```
<html>
<head>
<title>Internet and WWW How to Program Headers</title>
</head>
<body>
<h1>Level 1 Header</h1>
<h2>Level 2 header</h2>
<h3>Level 3 header</h3>
<h4>Level 4 header</h4>
<h5>Level 5 header</h5>
<h6>Level 6 header</h6>
</body>
</html>
```

The output will be:-

As seen above Starting with <h1> which displays the most prominent font size and ending with <h6> the least prominent font size.

- **PARAGRAPH TAG**:- Paragraphs are defined with the <p> tag to separate paragraphs i.e adds an extra blank line before and after a paragraph tags

  Example: <p> This is the start of a very small paragraph. p>

It may use attributes like align (values can be left, right, center, justify). For example; This is a paragraph

- **LINE BREAK TAG:-**
  - The <br> tag forces a line break wherever we place it. It uses to start new line.
  - The <br> tag is an empty tag. It has no closing tag.
- **COMMENT IN HTML:-**
  - Used to enter a comment in HTML source code.
  - The comment will be ignored by the browser.

  Example : <!-- This is a comment -->

<hr> taga is Used to delimit the parts by display a horizontal line.(devide between sections)

- **Horizontal tag:-**
  - Used to delimit the parts by display a horizontal line.
  - Written as <hr>

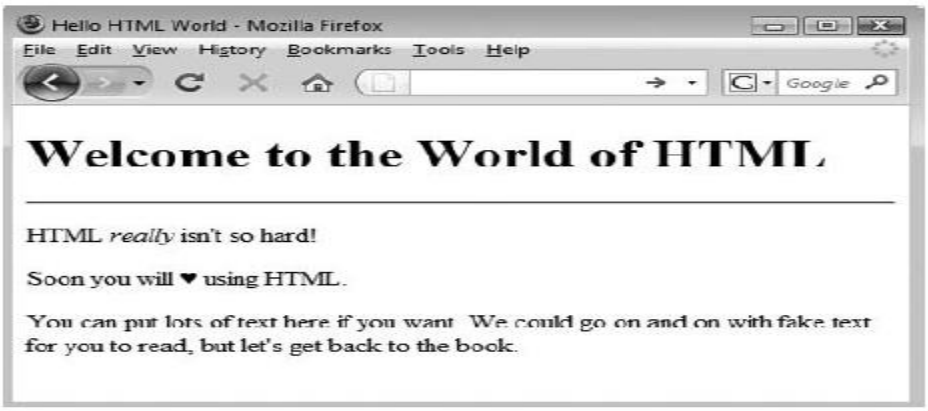
Horizontal Rule

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

The horizontal rule does not have a closing tag. It takes attributes such as align and width. For instance:

This Code	Would Display
<hr width="50%" align="center">>	

In html, horizontal tag may have attribute with not value as exceptional. But other attributes should have values. Example:- **<hr noshade> or <hr noshade="noshade" />** indicates that there should be no shading applied to this horizontal rule. Example:- write an html code that enables to display the following sample webpage:



See the following HTML tags and their descriptions:-

Tag	Description
<abbr>	Defines an abbreviation
<acronym>	Defines an acronym
<address>	Defines an address element
<cite>	Defines a *citation*
<code>	Defines `computer code` text
<blockquote>	Defines a long quotation
<del>	Defines text
<dfn>	Defines a *definition* term
	Defines *emphasized* text
<ins>	Defines inserted text
<kbd>	Defines keyboard text
<pre>	Defines preformatted text
<q>	Defines a short quotation
<samp>	Defines sample computer code
	Defines **strong** text
<var>	Defines a *variable*

Tag	Description
<b>	Defines **bold** text
<big>	Defines big text
<i>	Defines *italic* text
<small>	Defines small text
<sup>	Defines ^{superscripted} text
<sub>	Defines $_{\text{subscripted}}$ text
<tt>	Defines teletype text
<u>	Deprecated. Use styles instead

The <tag> tells the browser to do something, while the attribute tells the browser how to do it

Character tags like and produce the same physical display as <b> and <i> but are more uniformly supported across different browsers.

2.3. HTML ATTRIBUTES

An attribute is used to define the characteristics of an element and is placed inside the element's opening tag. The <tag> tells the browser to do something, while the attribute tells the browser how to do it. For instance, if we add the bgcolor attribute, we can tell the browser that the background color of the page should be blue, like this: <body bgcolor="blue">.

All attributes are made up of two parts: a **name** and a **value**.

- The name is the property that wanted to be set. For example, the <font> element in the example carries an attribute whose name is face, which we can use to indicate which typeface one want the text to appear in.
- The value is what we want the value of the property to be. The first example was supposed to use the Arial typeface, so the value of the face attribute is Arial

  <font face="arial" color="green">

Many HTML tags have a unique set of their own attributes. Right now we want to focus on a set of generic attributes that can be used with just about every HTML Tag in existence.

Core Attributes: The four core attributes that can be used on the majority of HTML elements (although not all) are: id, title, class and style.

I. **The id Attribute**: The **id** attribute can be used to uniquely identify any element within a page (or style sheet).

There are two primary reasons that we might want to use an id attribute on an element:

- If an element carries an id attribute as a unique identifier it is possible to identify just that element and its content.

- If there are two elements of the same name within a Web page (or style sheet), the id attribute uses to distinguish between elements that have the same name.

  This para explains what is HTML

  This para explains what is Casecading Style Sheet

II. Title Attributes: - The title attribute gives a suggested title for the element. They syntax for the title attribute is similar as explained for id attribute. The behavior of this attribute will depend upon the element that carries it, although it is often displayed as a tooltip or while the element is loading.

<h4 title="Hello HTML!">Titled Heading Tag Example</h4>

III. The class Attribute: The class attribute is used to associate an element with a style sheet, and specifies the class of element. For example: class="className1 className2 className3"

IV. The style Attribute: The style attribute allows to specify CSS/ casecade style sheet, we discussed in chapter 3, rules within the element. For example: Some text...

Note that:-

- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**
- Attributes and attribute values are case-insensitive.
- Attributes come in name/value pairs like: **name="value",** but not always
- Attribute values better be enclosed in double quotes.

The behavior of title attribute will depend upon the element that carries it, although it is often displayed as a tooltip or while the element is loading.

Attributes of Body tag

The attributes you can set for the page in the BODY tag are as follows:

You can set page-level properties in the BODY tag. Some of these are listed in the table below

Attribute	Description & Sample values
BACKGROUND	Background image – the specified image will be tiled across the page. If you do not want the image to tile, you can try to size the image (using a graphics package such as PhotoShop) so that it fills one page e.g. <BODY background="bgImage.gif">
BGCOLOR	Background colour – colours in HTML can be specified by name – but the name must be one of the set of reserved names (red, blue, green etc – see the HTMLib Reference for a full list - navigate to the section about the BODY tag, then find the section about BGCOLOR). Alternatively, a colour can be specified by a value. The value uses a combination of three pairs of *hexadecimal* numbers. Each pair represents the amount of red, green and blue in the colour. The value must be preceded by a #. For example, black is #000000, white is #ffffff.

	See section Error! Reference source not found. (below) about Using Colours in Web Pages for more information. e.g. <BODY bgcolor=”red”> or <BODY bgcolor=”#
TEXT	Text colour – specifies the colour for all text that does not have a colour specified otherwise. e.g. <BODY text=”green”>
LINK, **VLINK**, **ALINK** (ALINK only applies on Netscape browsers)	Link colours – the colours for links, visited links and active links can be specified (a link is active while it is being clicked on – this colour will only apply on Netscape browsers) <BODY link="red" vlink="green" alink="blue">
TOP MARGIN, **LEFT MARGIN**, **RIGHT MARGIN**, **BOTTOMMARGIN**	Setting the margins – the left, right, top and bottom margins of the page can be specified, in pixels; the margins are relative to the sides of the browser window. It is usual to set the top and left margins so that the page does not appear to be flush with the top left corner of the browser <BODY top margin="30" bottom margin="30" left margin="30" right margin="30">

Add text here

Example: - From <body bgcolor=”green”> sample example, bgcolor is the name of attribute, green is the value of the attribute.

- **Bgcolor** :- The bgcolor attribute specifies a background-color for an HTML page. The value of this attribute can be a hexadecimal number, an RGB value, or a color name: <body bgcolor="#000000">
<body bgcolor="rgb(0,0,0)"> <body bgcolor="black"> The lines above all set the background-color to black
- **Background: -** The background attribute can also specify a background-image for an HTML page. The value of this attribute is the URL of the image you want to use. If the image is smaller than the browser window, the image will repeat itself until it fills the entire browser window. See the following:-
  - <body background="clouds.gif"> <body <body
  - <background="http://profdevtrain.austincc.edu/html/graphics/clouds.gif">

vi. Text=”,… ”

- To determine the text color for a webpage. The value of the text attribute can be red, green, blue etc.

vii. Anchor tag attributes: anchor tag, <a>, uses to make hyperlink.

- **link**=”…”
  - To determine the color of hyperlinks on a webpage.
- **vlink**=”…”
  - To determine the color of visited hyperlinks on a webpage
- **alink**= “ ”
  - To determine the color of active hyperlinks on a webpage

There are HTML tags that are used to defines a lot of elements for formatting output, like **bold**, **italic text**, **bold-italic**, **superscript**, **subscrip**t, **small text**, **strong** and so on.

Example:-

Tag	Description
<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<s>	Deprecated. Use <del> instead
<abbr>	Defines an abbreviation
<acronym>	Defines an acronym
<address>	Defines an address element
<bdo>	Defines the text direction
<q>	Defines a short quotation
<cite>	Defines a citation
<dfn>	Defines a definition term

V. **Text direction attribute: - use <bdo dir=value**>. It uses to demonstrate how to change the text direction. The value of **dir** attribute can be **rtl** or **ltr.**

```
<html>
<body>
<p> If your browser supports bi-directional override (bdo), the next line will be written from the right
to the left (rtl):</p>
<bdo dir="rtl">Here is some Hebrew text</bdo>
</body>
</html>
```

VI. **Quotations**:- This example demonstrates how to handle quotations

```
<html>
<body>
<q> This is a long quotation</q>
<body>
```

```
</html>
```

VII. **Deleted and inserted text: - These are tags that are used to** mark a text that is deleted or inserted to a document. See the following example:

```
<html><body>
<p>a dozen is <del>twenty</del>
<ins>twelve</ins> pieces</p>
<p> Most browsers will overstrike deleted text and underline inserted text.</p>
<p>Some older browsers will display deleted or inserted text as plain text.</p>
</body></html>
```

VIII. **Abbreviations and acronyms: -** These tags use for handle an abbreviation or an acronym as written below.

```
<html><body>
<abbr title="United Nations">UN</abbr><br>
<acronym title="World Wide Web">WWW</acronym>
</body></html>
```

IX. **HTML Character Entities**: - Some characters like the < character, have a special meaning in HTML, and therefore cannot be used in the text. To display a less than sign (<) in HTML, we have to use a character entity.

A character entity has three parts: an ampersand (&), an entity name or a # and an entity number, and finally a semicolon (;).

To display a less than sign in an HTML document we must write: **<** or **<**

- Advantage of using a name: name is easier to remember.
- Disadvantage: not all browsers support the newest entity names, while the support for entity numbers is very good in almost all browsers.

Note That:- The entities are case sensitive

X. **Non-breaking Space: -** The most common character entity in HTML is the non-breaking space. Normally HTML will truncate spaces in the text. If we write 10 spaces in the text, HTML will remove 9 of them. To add spaces to text, use the   character entity. The following shows the description of lists of entity and non-breaking spaces

A character entity has three parts: an ampersand (&), an entity name or a # and an entity number, and finally a semicolon (;).

The entities are case sensitive

The Most Common Character Entities:

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&
"	quotation mark	"	"
'	apostrophe	' (does not work in IE)	'

Some Other Commonly Used Character Entities:

Result	Description	Entity Name	Entity Number
¢	cent	¢	¢
£	pound	£	£
¥	Yen	¥	¥
€	euro	€	€
§	section	§	§
©	copyright	©	©
®	registered trademark	®	®
×	multiplication	×	×
÷	division	÷	÷

A character entity has three parts: an ampersand (&), an entity name or an entity number, and finally a semicolon (;). The & means we are beginning a special character, the; means ending a special character and the letters in between are sort of an abbreviation for what it's for.

2.4. HTML Links/ Hyperlinks

Links are extremely useful for bringing the user to another page that may be within website, or may be another website that may useful to others. HTML uses a hyperlink to link to another document on the Web. A hyperlink (or link) is a word, group of words, or image that can be clicked on to jump to a new document or a new section within the current document. To create a link to another document, the hyperlink attribute uses the "href" attribute.

An anchor can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc. The HTML code for a link is simple, written as follow: - <a href="*url*">*Link text*</a>.

The <a> tag is used to create an anchor or link to link from, the href attribute is used to tell the address of the document or page we are linking to, and the words between the open and close of the anchor tag will be

displayed as a hyperlink. When we click on the displayed linker word(s), the browser will jump to the actual documents, URL. As shown from above hyperlink syntax,

- **href:** Means **H**ypertext **REF**erence. The href part is the **attribute**, with the location of the other page as its value.
- The href attribute specifies the destination of a link.
- The href or the name attribute must be present in the <a> tag.
- The href attribute is supported in all major browsers.

href: Means Hypertext REFerence

The name attribute is used to create a named anchor. When using named anchors we can create links that can jump directly to a specific section on a page, instead of letting the user scroll around to find what he/she is looking for. Unlike an anchor that uses href, a named anchor doesn't change the appearance of the text (unless we set styles for that anchor) or indicate in any way that there is anything special about the text. For example vlink and alink attributes.

To link directly to the top section, add a # sign and the name of the anchor to the end of a URL as:-

This Code	Would Display
<a href="http://profdevtrain.austincc.edu/html/10links.html#top">Back to top of page </a>	Back to top of page
A hyperlink to the top of the page from within the file 10links.html will look like this: <a href="#top">Back to top of page </a>	Back to top of page

In addition to href attribute, there are <a> tag attributes such as name and target attributes. An attribute called TARGET has values like "_blank" | "_parent" | "_self" | "_top" | *window name*

TARGET: - Controls where the new document will be displayed when the user follows a link. TARGET, the new document open in a new window, or if we are using frames, in another frame.

TARGET = "_blank"

- Used to opens the new document in a new window.
- < A HREF="newwindow.html" *TARGET="_blank"*>a new window</A>
- This value does not require the use of any frames.

TARGET = "_parent"

- Used in the situation where a frameset file is nested inside another frameset file.
- A link in one of the inner frameset documents which uses "_parent" will load the new document where the inner frameset file had been.
- For example, this anchor <A HREF="bigframe.html" *TARGET="_parent"*> bigframe </A>

TARGET = "_self"

- "_self" puts the new document in the same window and frame as the current document.
- "_self" works the same as if not used TARGET at all.
- For example <A HREF="selftarget.html" *TARGET="_self"*>next</A> page

TARGET = "_top"

- "_top" loads the linked document in the topmost frame... that is, the new page fills the entire window.
- For example,<A HREF="selftarget.html" *TARGET="_top"*>top</A>

N.B:- We can create an image hyperlink as follows:-

```
<a href="page1.html">
    <img src="image.gif" />
</a>
```

2.5. HTML Images

In HTML, images are defined with the <img> tag, to display the image on a page/ webpage. We use "src" attribute which stands for "source". The value of the src attribute is the URL of the image we want to display. Syntax for defining an image is **<img src="*url*" alt="*some_text*"/>**

The required alt attribute specifies an **alternate text** for an image, if the image cannot be displayed. The value of the alt attribute is an author-defined text: **<img src="pulpit.jpg" alt="Pulpit rock" width="304" height="228" >**. The height and width attributes are used to specify the height and width of an image.

Example:-

```
<html>
<body>
<h1 align="center">My First Web page</h1>
<p>Welcome to my first webpage.</p>
<p>By learning html, I'll be able to create web pages<br>
which I am of course.</p>
<!-- Who would have guessed how easy this would be -->
<p>
<img src="graphics/chef.gif" width="130" height="101" alt="Smiling Happy Chef"
align="center">
</p>
<p align="center">This is my Chef</p>
</body>
</html>
```

The most common options are:

- **SRC="FILENAME"** - Specifies the SOURCE of the image; if the image file is in the same directory as the source document, all that is needed is the filename itself. Note: Don't link to another page for an image - copy the image to the file space and link to it from there.
- **HEIGHT="NUMBER"** - Specifies that the height of the image should be NUMBER pixels
- **WIDTH="NUMBER"** - Specifies that the width of the image should be NUMBER pixels. If no dimensions are specified for the image, the browser will insert the image exactly as is, at the same size as developers would view it in a picture editor on computer. The picture will load slightly faster if you specify the height and width so that the browser does not have to determine it.
- **ALT="DESCRIPTION"** - Specifies a description of the image. The ALT option for the IMG tag is useful for those whose browsers cannot support in-line images, or have their images turned off in their browsers; instead of seeing the image, they'll see the description of the image that we gave them in the ALT tag.
- **BORDER="NUMBER"** - Specifies the width of the image's border, in pixels.
- **Hspace and vspace** also used to assign the horizontal and vertical spacing between images

Example:- An image tag could look like: <img src = "filename" height = "number" width = "number" alt = "description" border = "number"> like <img vspace="10" hspace="30" src="freenetlogo.gif" WIDTH="281" HEIGHT="108">

2.6. VIDEO/ Audio TAGS:-

To insert video, use a **<video>** tag and set its **src** attribute to a local or remote URL containing a playable movie. The **play black** can be displayed using controls by including the "**controls"** attribute, as well as set the dimensions of the movie to its natural size:-

```
<video src=http://htmlref.com/ch2/html_5.mp4 width="640" height="360" controls>
<Strong>HTML5 video element not supported</strong>
</video>
```

HTML's **audio** element is quite similar to the **video** element. The element should support common sound formats such as WAV files:

```
<audio src=http://htmlref.com/ch2/music.wav controls></audio>  or
<audio>
  <bgsound src=http://htmlref.com/ch2/music.wav controls>
</audio>
```

A complete example is shown here:

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

```
<h1>Simple Audio Examples</h1>
<h2>wav Format</h2>
<video src="a.mp4" controls/>
<audio src="http://htmlref.com/ch2/music.wav" controls></audio>
<h2>ogg Format</h2>
<audio src="http://htmlref.com/ch2/music.ogg" controls></audio>
<h2>Multiple Formats and Fallback</h2>
<audio controls autobuffer autoplay>
<source src="http://htmlref.com/ch2/music.ogg" type="audio/ogg">
<source src="http://htmlref.com/ch2/music.wav" type="audio/wav">
<!--[if IE]>
<bgsound src="http://htmlref.com/ch2/music.wav">
<![endif]-->
</audio>
</body>
</html>
```

The `<embed>` tag in HTML is used to embed external content such as images, videos, audio files, or other interactive content into an HTML document. It allows you to include multimedia elements within your web page. However, it's worth noting that the usage of the `<embed>` tag has been deprecated in HTML5, and the preferred way to include multimedia content is using the `<video>` and `<audio>` tags.

- Audio and video can also displayed by a tag called <**embed**>. It is written as <embed src=url>…<embed>

2.7. HTML Framesets

With frames, it is possible to display more than one Web page in the same browser window. It uses to partition a window into independent parts (frames) and used to display different documents in different frame with single interface.

The <frameset> tag defines how to divide the window into frames. Each frameset defines a set of rows or columns. The values of the rows/columns indicate the amount of screen area each row/column will occupy.

Common frameset tags are:

Tag	Description
<frameset>	Defines a set of frames, container tag type
<frame>	Defines a sub window (a frame). It is empty tag type. Used to display document.

<frame>empty tag type. Used to display document.

Two types of frameset:-

- **Vertical frameset:-**demonstrates how to make a vertical frameset with different documents.

```
<html>
<frameset cols="25%, 50%, 25 %">
<frame src="framea.htm">
```

```
<frame src="frameb.htm">
<frame src="framec.htm">
</frameset>
</html>
```

- **Horizontal frameset: -** demonstrates how to make a horizontal frameset with different documents.

```
<html><frameset rows="25%,50%,25%">
<frame src="framea.htm">
<frame src="frameb.htm">
<frame src="framec.htm">
</frameset></html>
```

Frames can divide the screen into separate windows. Each of these windows can contain an HTML document. A file that specifies how the screen is divided into frames is called a frameset. The content model says that the **<frameset>** tag contains one or more **<frame>** tags, which are used to indicate the framed contents. A **<frameset>** tag also might contain a **<noframes>** tag, whose contents will be displayed by browsers that do not support frames

There are few drawbacks of frames:-

- Some browsers do not print well from framesets.
- Some smaller devices cannot cope with frames, often because their screen is not big enough to be divided up.
- Some time the page will be displayed differently on different computers due to different screen resolution.
- The browser's *back button* might not work as the user hopes.
- There are still few browsers who do not support frame technology.

<frameset> tag also might contain a <noframes> tag, whose contents will be displayed by browsers that do not support frames

If needed to make a homepage that uses frames, we should use the followings:

- Make an HTML document with the frameset
- Make the normal HTML documents that should be loaded into each of these frames.

When a frameset page is loaded, the browser automatically loads each of the pages associated with the frames. Each HTML document is called a frame, and each frame is independent of the others.

The frameset element holds two or more frame elements. Each frame element holds a separate document. The frameset element states only HOW MANY columns or rows there will be in the frameset and it uses <frameset> tag. The <frame> tag defines one particular window (frame) within a frameset. In the example below we have a frameset with two columns. The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The document "framea.htm" is put into the first column, and the document "frameb.htm" is put into the second column:

```
<frameset cols="25%,75%">
  <frame src="framea.htm" />
  <frame src="frameb.htm" />
</frameset>
```

Note:

- The frameset column size can also be set in pixels (cols="200,500"), and one of the columns can be set to use the remaining space, with an asterisk (cols="25 %,*").
- No use the body element together with the frameset element. However, if <noframes> tag is added containing some text for browsers that do not support frames, it is possible to enclose the text in a body element.

Attributes of frame

Attribute	Value	Description
frameborder	0 or 1	Specifies whether or not to display a border around a frame
marginheight	Pixels	Specifies the top and bottom margins of a frame
marginwidth	Pixels	Specifies the left and right margins of a frame
noresize	Noresize	Specifies that a frame cannot be resized
scrolling	Yes, no auto	Specifies whether or not to display scrollbars in a frame
src	URL	Specifies the URL of the document to show in a frame

Attributes of frameset

Attribute	Value	Description
cols	Pixels, % or *	Specifies the number and size of columns in a frameset
rows	Pixels, % or *	Specifies the number and size of rows in a frameset

Nested Frames

Framesets may be nested to any level. In the following example, the outer FRAMESET divides the available space into three equal columns. The inner FRAMESET then divides the second area into two rows of unequal height.

Example: 1

```
<FRAMESET cols="33%, 33%, 34%">
   <frame src=”one.html”>
```

```
  <FRAMESET rows="40%, 50%">
    <frame src=”two.html”>
    <frame src=”three.html”>
  </FRAMESET>
<frame src=”four.html”>
</FRAMESET>
```

Example 2:-

```
<HTML>
<HEAD>
<TITLE>A simple frameset document</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
 <FRAMESET rows="100, 200">
    <FRAME src="contents_of_frame1.html">
    <FRAME src="contents_of_frame2.gif">
 </FRAMESET>
 <FRAME src="contents_of_frame3.html">
 <NOFRAMES>
    <P>This frameset document contains:
    <UL>
      <LI><A href="contents_of_frame1.html">Some neat contents</A>
      <LI><IMG src="contents_of_frame2.gif" alt="A neat image">
      <LI><A href="contents_of_frame3.html">Some other neat contents</A>
    </UL>
 </NOFRAMES>
</FRAMESET>    </HTML>
```

Note: the more frames are nested; it will make it complicated hence difficult manageable.

NOFRAMES:-

The NOFRAMES element specifies content that should be displayed only by user agents that do not support frames or are configured not to display frames. User agents that support frames must only display the contents of a NOFRAMES declaration when configured not to display frames. User agents that do not support frames must display the contents of NOFRAMES in any case. NOFRAMES may be used at the end of the FRAMESET section of the document.

a **<body>** tag should be found for browsers that do not support frames. A visual representation of this idea is shown here:

Example1:-
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
 <HTML>
 <HEAD>
 <TITLE>A frameset document with NOFRAMES</TITLE>
 </HEAD>
 <FRAMESET cols="50%, 50%">
   <FRAME src="main.html">
   <FRAME src="table_of_contents.html">
   <NOFRAMES>
   <P>Here is the <A href="main-noframes.html">
          non-frame based version of the document.</A>
   </NOFRAMES>
 </FRAMESET>
 </HTML>
```

Example 2:-

```
<head>
  <meta http-equip="Content-Type"
  content="text/html; charset=utf-8">
  <title>Page Title</title>

</head>

<frameset>
<frame />
<frame />

</frameset>

<noframes>
<body>

</body>
</noframes>

</html>
```

Head contains info about page

Root HTML element encloses entire doc

Frameset

Body

Noframes compatibility

HTML 4 and 4.01 (frameset DTD) XHTML 1.0 (frameset DTD)	Firefox 1+, Internet Explorer 3+, Netscape 2+, Opera 4+, Safari 1+

IFRAME - **Inline Frames**

The IFRAME element allows authors to insert a frame within a block of text. Inserting an inline frame within a section of text is much like inserting an object via the OBJECT element: they both allow to insert an HTML document in the middle of another, they may both be aligned with surrounding text, etc.

The information to be inserted inline is designated by the **src** attribute of this element. The *contents* of the **IFRAME** element, on the other hand, should only be displayed by user agents that do not support frames or are configured not to display frames.

The <iframe> tag is not used within a <frameset> tag. Instead, it appears anywhere in the document. The <iframe> tag defines a rectangular region within the document in which the browser displays a separate document, including scrollbars and borders.

```
<body>
...other document content...
<iframe src="/html/menu.htm" width="75" height="200" align="right">
Your browser does not support inline frames. To view this
<a href="/html/menu.htm">document</a> correctly, you'll need
a copy of Internet Explorer or the latest Netscape Navigator.
</iframe>
...subsequent document content...
```

```
</body>
```

For user agents that support frames, the following example will place an inline frame surrounded by a border in the middle of the text.

```
<IFRAME src="foo.html" width="400" height="500"
        scrolling="auto" frameborder="1">
[Your user agent does not support frames or is currently configured
not to display frames. However, you may visit
<A href="foo.html">the related document.</A>]
</IFRAME>
```

2.8. HTML Tables

Tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, other tables, etc. Header information in a table is defined with the <th> tag. The text in a th element will be **bold** and **centered**.

Tags	Description
<Table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns
<col>	Defines the attribute values for one or more columns in a table

The text in a th tag element will be bold and centered.

Example:

```
<table border="1">
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td bgcolor=green>row 1, cell 1</td>
<td bgcolor=red>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
```

```
<td>row 2, cell 2</td>
</tr>
</table>
```

If no specify a border attribute to the table, it will be displayed without any borders

HTML code above looks in a browser:

Header 1	**Header 2**
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

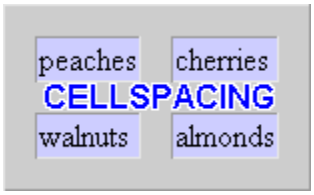
Tables' attribute

Border: - If no specify a border attribute to the table, it will be displayed without any borders. Sometimes this can be useful, but most of the time, we want the borders with value of some number.

<table border="n">

Headings in a table: demonstrates how to display table headers. <html><body><h4> **Table headers:**

- **Non break space**: to use " " to handle cells that have no content.
- **Table with a caption**: demonstrates a table with a caption
- **Cell padding**: to create more white space between the cell content and its borders.
- **CELLSPACING**: controls the space between table cells. Although there is no official default, browsers usually use a default of 2.



- **CELLPADDING:** sets the amount of space between the contents of the cell and the cell wall. The default is 1. CELLPADDING is usually more effective than CELLSPACING for spreading out the contents of the table

Attributes of table: the attributes of a table will be applied on the whole table element which include one or more rows (<tr>), header cells (<th>) or data cells (<td>)

Attribute	**Value**	**Description**
align	left center	Specifies the alignment of a table according to surrounding

	right	text
bgcolor	rgb(x,x,x) #xxxxxx colorname	Specifies the background color for a table
border	Pixels	Specifies the width of the borders around a table
cellpadding	Pixels	Specifies the space between the cell wall and the cell content
cellspacing	Pixels	Specifies the space between cells
width	pixels %	Specifies the width of a table

Table 2.1 attributes of table

Attributes of rows and cells: this attributes will be applicable only to the header cell or the data cell if it is used with <th> or <td> tag while it will affect the whole content of the row if it is used by the <tr> tag.

Attribute	**Value**	**Description**
align	left right center justify char	Aligns the content in a cell
bgcolor	rgb(x,x,x) #xxxxxx colorname	Specifies a background color for a cell
colspan	Number	Specifies the number of columns a cell should span
height	pixels %	Sets the height of a cell
nowrap	Nowrap	Specifies that the content inside a cell should not wrap
rowspan	Number	Sets the number of rows a cell should span
width	pixels %	Deprecated. Use styles instead. Specifies the width of a cell

valign	top middle bottom	Vertical aligns the content in a table row

Table attributes of rows and cells

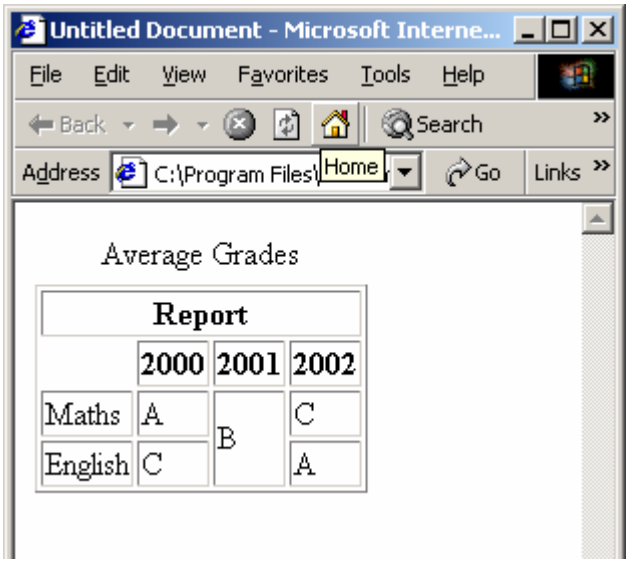
COLSPAN and ROWSPAN

Table cells can span across more than one column or row. Any cell (<th> or <td> tag) can have a rowspan="integer " or colspan=" integer" attribute to “span” more than one cell of the table, that is, place that element's content into an area that would normally be occupied by two or more cells.

The attributes COLSPAN ("how many across") and ROWSPAN ("how many down") indicate how many columns or rows a cell should take up.

- COLSPAN number of columns to cover
- ROWSPAN: number of rows to cover
- WIDTH: width of cell
- HEIGHT: height of cell
- BGCOLOR: color of the background
- BORDERCOLOR: color of border around the table
- ALIGN: horizontal alignment of cell contents
- VALIGN: vertical alignment of cell contents
- **VSPACE:** vertical space between table and surrounding text
- **HSPACE:** horizontal space between table and surrounding text
- **RULES**: if there should be internal borders

Example: - write html code to display the following web page



Nesting Tables

Tables may be nested to produce interesting effects, such as the following example: Because the TABLE element must have an end tag, no particular tricks exist for nesting tables. we may include a table within another table's cell or another table's row

Table nest example:

```
<TABLE border="1" cellspacing="2"
cellpadding="2">
<TR>
<TD colspan="3">A1
<TR>
<TD>B1
<TD>B2
<TD rowspan="2">B3
<TR>
<TD colspan="2">C1
</TABLE>
```

To add a title to a table, place the caption text between <caption> and </caption> tags just after the <table> tag.

To add a title to a table, place the caption text between <caption> and </caption> tags just after the <table> tag. The align="bottom" attribute in the <caption> can be added. Table tag may have width attribute with value in persent or inch as <table width="75 %">…</table> or <table width="5.5in">...</table>

```
<html>
<body>
<TABLE border="5" cellspacing="2"
cellpadding="5" bgcolor=red>
<caption>Ethiopian table</caption>
<tr>
<th colspan="3" align="center">Lifespan</th>
<th>top</th>
</tr>
<tr>
<td>Name
```

```
<td>Born
<td>hhhh
<td>Died
</tr>
<tr><td>25
<td>1900
<td> 
<td>1971
</tr>
</table>
</body>
</html>
```

2.9. HTML Lists

It uses to list down sets of documents with sequence of numbers of bullets of combination of the two. The followings are html List Tags

Tag	Description
<ol>	Defines an ordered list
<ul>	Defines an unordered list
<li>	Defines a list item
<dl>	Defines a definition list
<dt>	Defines a definition term
<dd>	Defines a definition description
<dir>	Deprecated. Use <ul> instead
<menu>	Deprecated. Use <ul> instead

Different types of html lists:-

- **Unordered Lists: - An** unordered list starts with the <ul> tag. Each list item starts with the <li> tag. The list items are marked with bullets (typically small black circles). The "type" attribute can be used to specifies the style of the bullet points of the list items, its value includes disc, square and circle.

  Example:-

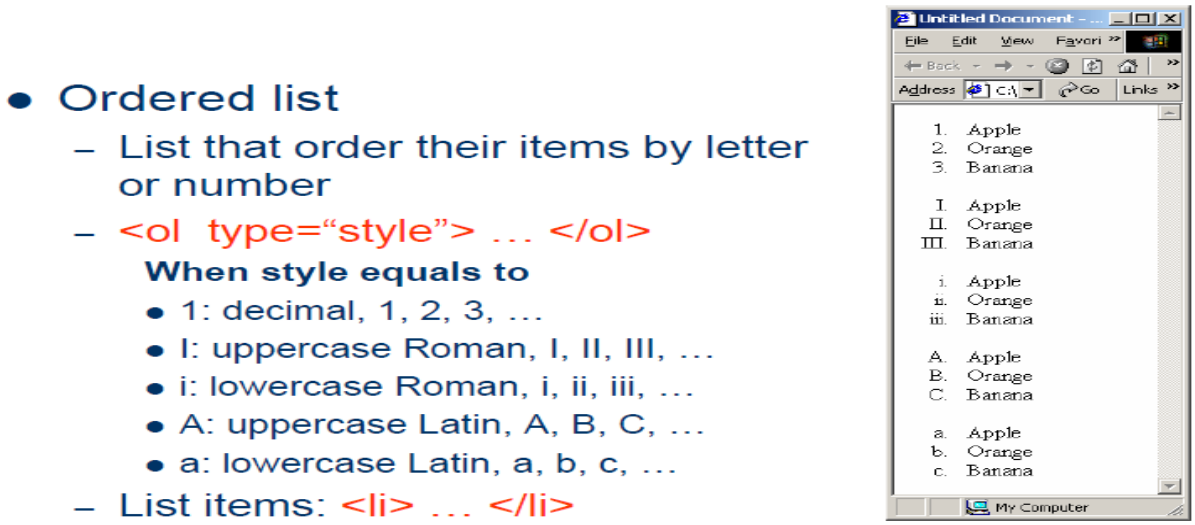
```
<html>
<body>
<h4>Disc bullets list:</h4>
<ul type="disc">
<li>Apples</li>
```

```
<li>Bananas</li>
<li>Lemons</li>
<li>Oranges</li <html><body>
```

- **Order List:-** An orderlist use <ol> tag type with attribute of "1", "a", "i" etc which are grouped under number types.

```
<html>
<body>
<h4>Numbered list:</h4>
<ol type=A>
<li>Apples</li>
<li>Bananas</li>
<li>Lemons</li>
<li>Oranges</li>  </ol>
```

Each list item starts with the <li> tag to lists down with sequences



- **Definition Lists :-** A definition list is **not** a list of items. This is a list of terms and explanation of the terms. A definition list starts with the <dl> tag. Each definition-list term starts with the <dt> tag. Each definition-list definition starts with the <dd> tag.

A definition list is not a list of items. This is a list of terms and explanation of the terms

```
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
 <dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
```

Here is how it looks in a browser: Coffee Black hot drink Milk White cold drink Inside a definition-list definition (the <dd> tag) you can put paragraphs, line breaks, images, links, other

2.10. HTML FORM

The <form> tag is used to create an HTML form for user input. A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select menus, textarea, fieldset, legend, and label elements.

HTML forms are used to pass data to a server. An HTML form is a section of a document containing normal content, markup, special elements called controls (checkboxes, radio buttons, menus, etc.), and labels on those controls. Users generally "complete" a form by modifying its controls (entering text, selecting menu items, etc.), before submitting the form for further processing (e.g., to a Web server, to a mail server, etc.). Forms are a vital tool for the webmaster to receive information from the web server, such as: their name, email address, credit card, etc. A form will take input from the viewer and depending on needs; we may store that data into a file, place an order, gather user statistics, register the person to our web forum, or maybe subscribe them to our weekly newsletter.

The <form> tag is used to create an HTML form:

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

The most important form element is the input element. The input element is used to select user information. An input element can vary in many ways, depending on the type attribute. An input element can be of type text field, checkbox, password, radio button, submit button, and more. Elements inside a form are introduced by the <input> tag.

The <input> tag may use the following attributes as values:-

- type="hidden" name="variable name" value="value that sends to the server"
- type ="text" name="" value ="" size="25"
- type ="submit" value =""
- type ="reset" value =""
- type ="checkbox" value ="" name=""

Working with Text Fields and Passwords

Text Field

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

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

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

The output will look as:

First name:

Last name:

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

Password

Written as <input type="password" /> defines a password field. See example as

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

The output looks as:

Password: *********

Note: the characters entered in to the password field are masked.

Using Buttons, Checkboxes and Selection Lists

Radio Buttons:-

- Radio buttons are a popular form of interaction. we may have seen them on quizzes, questionnaires, and other web sites that give the user a multiple choice question. Below are a couple attributes you should know that relate to the radio button.
  - **Value:** specifies what will be sent if the user chooses this radio button. Only one value will be sent for a given group of radio buttons.
  - **Name**: defines which set of radio buttons that it is a part of.
- **Written as** <input type="radio" /> defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices: see example

```
<form>
<input type="radio" name="sex" value="male" /> Male<br />
<input type="radio" name="sex" value="female" /> Female
</form>
```

Checkboxes

- Check boxes allow for multiple items to be selected for a certain group of choices. The check box's name and value attributes behave the same as a radio button.

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

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

Example:-

```
<p>Please select the courses you studied in college:
<br />
<input type="checkbox" name="courses" value="bio" checked="checked" />Biology
<br />
<input type="checkbox" name="courses" value="chem" />Chemistry
<br />
<input type="checkbox" name="courses" value="anat" />Anatomy
<br />
<input type="checkbox" name="courses" value="phys" />Physics
<br />
<input type="checkbox" name="courses" value="psych" />Psychology
<br />
<input type="checkbox" name="courses" value="eng" />English
<br />
<input type="checkbox" name="courses" value="bowl" />Bowling
</p>
```

Selection lists

- Drop down menus are created with the <select> and <option> tags. <select> is the list itself and each <option> is an available choice for the user.

```
<select name="degree">
<option>Choose One</option>
<option>Some High School</option>
<option>High School Degree</option>
<option>Some College</option>
<option>Bachelor's Degree</option>
<option>Doctorate</option>
<input type="submit" value="Email Yourself">
</select>
```

Drop down menus are created with the <select> and <option> tags

size attribute Specifies the number of visible options in a drop-down list

Example 2:-

Favorite language:

<SELECT NAME="language">

```
<OPTION VALUE="c">C
<OPTION VALUE="c++">C++
<OPTION VALUE="java" SELECTED>Java
<OPTION VALUE="lisp">Lisp
<OPTION VALUE="perl">Perl
<OPTION VALUE="smalltalk">Smalltalk
</SELECT>
```

The followings are some of the common select attributes:-

Attribute	Value	Description
disabled	Disabled	Specifies that a drop-down list should be disabled
multiple	Multiple	Specifies that multiple options can be selected
name	Name	Specifies the name of a drop-down list
size	Number	Specifies the number of visible options in a drop-down list

Text Area Tags:-

- It uses to create an area in our form, where visitors can enter longer answers such as entering an abstract or essay. Here, visitors can enter up to 32, 700 characters in a text area.
- It is aritten as <textarea>…..</textarea> to create input area
- The text area tag has several attributes such as name, rows, cols and wrap.
- See the following example:-

```
<textarea name="abstract" rows="10" cols="70" wrap="wrap">
</textarea>
```

Where

The **Name** attribute allows you to identify the field when you view the information. It is also often the field header associated with the text in the database.

The **Rows** and **Cols** attributes determine the size of the Text Area in the web page. The Rows attribute defines the height of the text box, while the Cols attribute specifies the width of the text box. If the user types in more text than fits in the area, vertical and/or horizontal scroll bars appear.

The **Wrap** attribute within the Text Area tag will cause the text to wrap when the end of the horizontal space has been reached. This can make input text easier for the viewer to read.

Example:-

```
<p>please comment the document what I sent before:-<br>
```

```
<textarea name="selfdescribition" rows="10" cols="80" wrap="wrap">
</textarea>
</p>
```

Example:-

```
<html>
<body>
<textarea name="abstract" rows="100" cols="170" wrap="wrap">
</textarea>
<input type="text" value="enteryour comment"/>
</body>
</html>
```

Example:-

```
<CENTER>
<P>
Enter some HTML:<BR>
<TEXTAREA NAME="HTML" ROWS=5 COLS=30>
Delete this text and replace
With some HTML to validate.
</TEXTAREA>
<CENTER>
```

To create a link between PHP scripts and HTML forms, we need to use the functionalities of buttons , and the attribute of the <form tag>, specifically "action" and "method"

Reset button

- The reset buton clears the form and returns all the settings to their defalt values. The attribute value is the text that will appear in the button
- Written as <input type="reset" value ="reset the form"/>

2.11. Linking HTML Forms with PHP Scripts

To create a link between PHP scripts and HTML forms, we need to use the functionalities of buttons, and the attribute of the <form tag>, specifically "action" and "method".

Buttons:

Submit buttons: When activated, a submit button submits a form. A form may contain more than one submi button.

Reset buttons: When activated, a reset button resets all controls to their initial values.

<input type="submit" /> defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute (how to write scripts will be discussed in the coming chapters).

Action and method:

The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="action.php" method="get">
Username: <input type="text" name="user" />
<input type="submit" value="Submit" />
```

Automating Processing: Info Forms and Emails

If we would like to provide our web site visitors with a simple way to contact us from our web site, but really don't want to display our email address, this HTML form code may be what we're looking for. We can create a simple form, as displayed below, to enable our visitors to send us comments, questions, product support requests, or whatever we'd like:-

```
<form  action=mailto:dummy1@yahoo.com, method="POST", enctype="text/plain">
Your Name: <input type="text" size="20" name="VisitorName"><br><br>
Your Comment: <textarea name="VisitorComment" rows="4" cols="20">
</textarea><br><br>
<input type="submit" value="Email This Form">
</form>
```

form action email keee gochuu dandeesa jechuudha

Example:-

```
<input type="hidden" name="title" value="Feedback" />
<p><label>Name:
        <input type= "text" name= "name" size="25" maxlength="30"/>
  </label>
 </p>
<input type= "submit" value="Submit your entries"/>
<input type= "reset" value="Clear Your Entries"/>
```

Example

2.12. Creating File upload

HTML Element: <INPUT TYPE="FILE" ...> (No End Tag). **Attributes:** NAME (required), VALUE (ignored), SIZE, MAXLENGTH, ACCEPT, ONCHANGE, ONSELECT, ONFOCUS, ONBLUR (nonstandard).

This element results in a filename textfield next to a Browse button. Users can enter a path directly in the textfield or click on the button to bring up a file selection dialog that lets them interactively choose the path to

a file. When the form is submitted, the *contents* of the file are transmitted as long as an ENCTYPE of multipart/form data was specified in the initial FORM declaration.

For multipart data, you also need to specify POST as the method type. This element provides a convenient way to make user-support pages, with which the user sends a description of a problem along with any associated data or configuration files.

ENCTYPE="multipart/ form-data file upload godhuuf ibsuu qabna

For multipart data, you also need to specify POST as the method type

It uses attributes as:-

- The NAME attribute identifies the text field to the server-side program.
- **VALUE**

For security reasons, this attribute is ignored; only the end user can specify a filename. Otherwise, a malicious HTML author could steal client files by specifying a filename and then using JavaScript to automatically submit the form when the page is loaded.

- **ONCHANGE, ONSELECT, ONFOCUS, and ONBLUR**

These attributes are used by browsers that support JavaScript to specify the action to take when the mouse leaves the textfield after a change has occurred, when the user selects text in the textfield, when the textfield gets the input focus, and when the textfield loses the input focus, respectively.

Example:-

```
<FORM ACTION="http://localhost:8088/SomeProgram"
ENCTYPE="multipart/form-data" METHOD="POST">
Enter data file below :<BR>
<INPUT TYPE="FILE" NAME="fileName">
<input type="submit" value="upload the file" />
</FORM>
```

- It uses to upload different files such as word and pdf formats
- See the following example:-

```
<html>
<body>
<form method="post" action="cgi-bin/upload.cgi" enctype="multipart/form-data">
<p><strong> file upload area</strong></p>
<p>please select the file to upload:<br/>
<input type="file" name="title" size="50"/>
<input type="submit" value="upload the file" />
</body>
</html>
```

Exercise: Test the output of the following codes

1.

```
<html>
<body>
  <form action="<?php $_PHP_SELF ?>" method="POST">
  FName: <input type="text" name="fname" />
  LName: <input type="text" name="lname" />
  GName: <input type="text" name="gname" />
  Department: <input type="text" name="dname" />
  College: <input type="text" name="cname" />
  Age: <input type="text" name="age" />
  <input type="submit" />
  </form>
</body>
</html>
```

2.

```
<html>
<head>
<title>Personal INFO</title>
</head>
<body>
<form method="post" action=”a.php" > // assume you have a.php in the folder
First Name:<input type="text" size="12" maxlength="12" name="Fname"><br />
Last Name:<input type="text" size="12" maxlength="36" name="Lname"><br />
Gender:<br />
Male:<input type="radio" value="M" name="gender"><br />
Female:<input type="radio" value="F" name="gender"><br />
Please choose type of residence:<br />
Steak:<input type="checkbox" value="Steak" name="food"/><br />
Pizza:<input type="checkbox" value="Pizza" name="food"/><br />
Chicken:<input type="checkbox" value="Chicken" name="food"/><br />
<textarea rows="5" cols="20" name="quote" wrap="physical">Enter your favorite
quote!</textarea><br />
```

```
        Select a Level of Education:<br />
        <select name="education">
        <option value="Jr.High">Jr.school</option>
        <option value="HighSchool">HighSchool</option>
        <option value="College">College</option>
        <option value="College">university</option>
        </select><br />
        Select your favorite time of day:<br />
        <select name="TofD">
        <option value="Morning">Morning</option>
        <option value="Day">Day</option>
        <option value="Night">Night</option>
        </select><br />
        <input type="radio" name="likeit" value="Yes" checked="checked" /> Yes
        <input type="radio" name="likeit" value="No" /> No
        <input type="radio" name="likeit" value="Not sure" /> Not sure</p>
        <select name="crew">
  <option value="xebrax">Xebrax</option>
  <option value="snertal">Snertal</option>
  <option value="gosny">Gosny</option>
</select><br>
        <input type="submit" value="submit" name="submit">
        </form>
          </body>
        </html>
```

the same with previouse chapter jump it

Chapter 2

2. Hyper Text Markup Language (HTML)

2.1. Introduction to HTML

What is HTML?

HTML is an acronym which stands for Hyper Text Markup Language.

Hyper Text: Hyper Text simply means "Text within Text". A text has a link within it, is a hypertext. Every time when you click on a word which brings you to a new webpage, you have clicked on a hypertext.

Markup language: A markup language is a programming language that is used to make text more interactive and dynamic. It can turn a text into images, tables, links etc.

- HTML is used to create web pages.
- HTML is widely used language on the web.
- We can create static website by HTML only.

What do you need to get started?

To get started HTML you need a computer installed with HTML editor and web browser.

- **HTML Editor:** HTML can be written and edited using many different editors like Dreamweaver and Visual Studio. However, in this course we use a plain text editor (like Notepad, Note pad ++) to edit HTML.
- **Web Browser:** such as Mozilla Firefox, Google Chrome, Opera etc.

2.2. HTML Basics, Elements and Attributes

An HTML document is made of many HTML tags and each HTML tag contains different content.

HTML Tags

HTML tags contain three main parts: opening tag, content and closing tag. But some HTML tags are unclosed tags. When a web browser reads an HTML document, browser reads it from top to bottom and left to right. HTML tags are used to create HTML documents and render their properties. Each HTML tags have different properties.

Syntax: <tag> content </tag>

- HTML tags are keywords surrounded by **angle brackets** like <html>
- HTML tags normally **come in pairs** like and
- 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**

Unclosed HTML Tags

Some HTML tags are not closed, for example br, hr, img etc.

- **
 Tag**: br stands for break line, it breaks the line of the code.
- **<hr> Tag**: hr stands for Horizontal Rule. This tag is used to put a line across the webpage.

HTML Documents = Web Pages

- HTML documents **describe web pages**
- HTML documents **contain HTML tags** and plain text
- HTML documents are also **called web pages**

The purpose of a web browser (like Internet Explorer or Firefox) is to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page:

```
<html><body><h1>My First Heading</h1> <p>My first paragraph.</p>
</body></html>
```

- The text between <html> and </html> describes the web page
- The text between <body> and </body> is the visible page content
- The text between <h1> and </h1> is displayed as a heading
- The text between and is displayed as a paragraph

Tag	Description
<head>	Defines information about the document
<title>	Defines the title of a document
<base>	Defines a default address or a default target for all links on a page
<link>	Defines the relationship between a document and an external resource
<meta>	Defines metadata about an HTML document
<script>	Defines a client-side script
<style>	Defines style information for a document

The HTML `<head>` element has nothing to do with HTML headings. The `<head>` element is a container for metadata. HTML metadata is data about the HTML document. Metadata is not displayed. The `<head>` element is placed between the `<html>` tag and the `<body>` tag:

Example

```
<!DOCTYPE html>
<html><head> <title>My First HTML</title>  <meta charset="UTF-
8"></head><body><h1>This is my first html page</h1></body></html>
```

Note: Metadata typically define the document title, character set, styles, links, scripts, and other meta information.

How to View HTML Source code?

:

- Right-click in an HTML page and select "View Page Source" (in Chrome) or "View Source" (in IE), or similar in other browsers. This will open a window containing the HTML source code of the page.

HTML Elements

An HTML element usually consists of a **start** tag and **end** tag, with the content inserted in between: `<tagname>Content goes here...</tagname>`

The HTML **element** is everything from the start tag to the end tag:

`<p>My first paragraph.</p>`

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
	My first paragraph.	

HTML elements with no content are called **empty elements**. Empty elements do not have an end tag, such as the <br> element (which indicates a line break).

Nested HTML Elements

HTML elements can be nested (elements can contain elements).

All HTML documents consist of nested HTML elements.

This example contains four HTML elements:

Example

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

The `<html>` element defines the **whole document**. It has a **start** tag <html> and an **end** tag </html>. The element **content** is another HTML element (the `<body>` element). The `<body>` element defines the **document body**. It has a **start** tag <body> and an **end** tag </body>. The element **content** is two other HTML elements (`<h1>` and `<p>`). The `<h1>` element defines a **heading**. It has a **start** tag <h1> and an **end** tag </h1>. The element **content** is: My First Heading. The `<p>` element defines a **paragraph**. It has a **start** tag and an **end** tag .

HTML Attributes

- All HTML elements can have **attributes**
- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**

The HTML element attribute has the following **syntax**:

```
<tagname attributename="property:value">
```

Example: style, href ,src are html attributes

- <a **href**="https://www.google.come">This is a link</a> - href(hypertext reference-source of file used by the tag) is used to specify a link address.
- <img **src**="img.jpg"> - src is used to specify filename of an image source.
- <img src="img.jpg" width="500" height="600"> - Images in HTML have a set of **size** attributes, which specifies the width and height of the image
- I am a paragraph - style attribute is used to specify the styling of an element. Such as color, font, size etc.

2.3. HTML Basic Elements function

HTML Headings

Headings are defined with the `<h1>` to `<h6>` tags. `<h1>` defines the most important heading. `<h6>` defines the least important heading.

Example

```
<h1>Heading 1</h1> <h2>Heading 2</h2> <h3>Heading 3</h3>
<h4>Heading 4</h4> <h5>Heading 5</h5> <h6>Heading 6</h6>
```

Note: Browsers automatically add some white space (a margin) before and after a heading.

HTML Horizontal Rules

The `<hr>` tag defines a thematic break in an HTML page, and is most often displayed as a horizontal rule. The `<hr>` element is used to separate content (or define a change) in an HTML page:

Example

```
<h1>This is heading 1</h1><p>This is some text.</p> <hr>
<h2>This is heading 2</h2><p>This is some other text.</p> <hr>
```

HTML Paragraphs

The HTML `<p>` element defines a **paragraph**:

Example

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

Note: Browsers automatically add some white space (a margin) before and after a paragraph.

HTML Line Breaks

The HTML `<br>` element defines a **line break**.

Use `<br>` if you want a line break (a new line) without starting a new paragraph:

Example

```
<p>This is<br>a paragraph<br>with line breaks.</p>
```

The `<br>` tag is an empty tag, which means that it has no end tag.

HTML Formatting Elements

Like Style attribute, HTML also defines special **elements** for defining text with a special **meaning**. HTML uses elements like `<b>` and `<i>` for formatting output, like **bold** or *italic* text.

Formatting elements were designed to display special types of text:

HTML and Elements

The HTML `<b>` element defines **bold** text, without any extra importance.

Example

This text is bold

The HTML `<strong>` element defines **strong** text, with added semantic "strong" importance.

Example

This text is strong

HTML <i> and Elements

The HTML `<i>` element defines *italic* text, without any extra importance.

Example

```
<i>This text is italic</i>
```

The HTML `<em>` element defines *emphasized* text, with added semantic importance.

Example

```
<em>This text is emphasized</em>
```

Note: Browsers display `<strong>` as `<b>`, and `<em>` as `<i>`. However, there is a difference in the meaning of these tags: `<b>` and `<i>` defines bold and italic text, but `<strong>` and `<em>` means that the text is "important".

HTML <mark> Element

The HTML `<mark>` element defines marked or highlighted text:

Example

```
<h2>HTML <mark>Marked</mark> Formatting</h2>
```

HTML <sub> Element

The HTML `<sub>` element defines $_{\text{subscripted}}$ text.

Example

```
<p>This is <sub>subscripted</sub> text.</p>
```

HTML <sup> Element

The HTML `<sup>` element defines ^{superscripted} text.

Example

```
<p>This is <sup>superscripted</sup> text.</p>
```

HTML Comments

Comment tags are used to insert comments in the HTML source code. You can add comments to your HTML source by using the following syntax:

<!-- Write your comments here -->

Notice that there is an exclamation point (!) in the opening tag, but not in the closing tag.

Note: Comments are not displayed by the browser, but they can help document your HTML source code.

With comments you can place notifications and reminders in your HTML:

Example

<!-- This is a comment --> This is a paragraph.

HTML Entities

Reserved characters in HTML must be replaced with character entities. Characters that are not present on your keyboard can also be replaced by entities. Some characters are reserved in HTML. If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags. Character entities are used to display reserved characters in HTML.

A character entity looks like this: &*entity_name*; OR &#*entity_number*;

To display a less than sign (<) we must write: **<** or **<**

Example: Non-breaking Space **( )**

A common character entity used in HTML is the non-breaking space: ** **

A non-breaking space is a space that will not break into a new line.

Two words separated by a non-breaking space will stick together (not break into a new line). This is handy when breaking the words might be disruptive.

Some Useful HTML Character Entities

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	<	<
>	greater than	>	>
&	Ampersand	&	&
"	double quotation mark	"	"
'	single quotation mark (apostrophe)	'	'
¢	Cent	¢	¢
£	Pound	£	£
¥	Yen	¥	¥
€	Euro	€	€
©	Copyright	©	©
®	registered trademark	®	®

Note: Entity names are case sensitive.

2.4. HTML links, images, tables and lists

HTML Links

Links are found in nearly all web pages. Links allow users to click their way from page to page. HTML links are hyperlinks. You can click on a link and jump to another document. When you move the mouse over a link, the mouse arrow will turn into a little hand.

Note: A link does not have to be text. It can be an image or any other HTML element.

HTML Links - Syntax

In HTML, links are defined with the `<a>` tag: <a href="*url of required web page*">*link text*</a>

Example: <a href="https://www.dmu.edu.et/">Visit our website</a>

The `href` attribute specifies the destination address (https://www.dmu.edu.et) of the link.

The **link text** is the visible part (Visit our website). Clicking on the link text will send you to the specified address.

Local Links

The example above used an absolute URL (a full web address). A local link (link to the same web site) is specified with a relative URL (without https://www....).

Example: <a href="vacancy.html">Vacancy</a>

HTML Link Colors

By default, a link will appear like this (in all browsers):

- An unvisited link is **underlined and blue**
- A visited link is **underlined and purple**
- An active link is **underlined and red**

You can change the default colors, by using CSS.

HTML Links - The target Attribute

The `target` attribute specifies where to open the linked document. The target attribute can have one of the following values:

- _blank - Opens the linked document in a new window or tab
- _self - Opens the linked document in the same window/tab as it was clicked (this is default)
- _parent - Opens the linked document in the parent frame
- _top - Opens the linked document in the full body of the window
- *framename* - Opens the linked document in a named frame

Tip: If your webpage is locked in a frame, you can use `target="_top"` to break out of the frame:

External pages can be referenced with a full URL or with a path relative to the current web page. This example uses a full URL to link to a web page:

Example: <a href="https://www.w3schools.com/html/default.asp">HTML tutorial</a>

This example links to a page located in the html folder on the current web site:

Example: <a href="/html/default.asp">HTML tutorial</a>

This example links to a page located in the same folder as the current page:

Example: <a href="default.asp">HTML tutorial</a>

HTML Images

Images can improve the design and the appearance of a web page. In HTML, images are defined with the `<img>` tag. The `<img>` tag is empty, it contains attributes only, and does not have a closing tag. The `src` attribute specifies the URL (web address) of the image:

<img src="*url of image*">

The alt Attribute

The `alt` attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader). The value of the `alt` attribute should describe the image.

Image Size - Width and Height

You can use the `style` attribute to specify the width and height of an image. The `width` and `height` attributes always define the width and height of the image in pixels.

Example: <img src="img_dmulogo.jpg" alt="dmu logo" style="width:500px;height:600px;">

Alternatively, you can use the `width` and `height` attributes:

<img src=" img_dmulogo.jpg" alt=" dmu logo " width="500" height="600">

HTML Audio Tag

HTML audio tag is used to define sounds such as music and other audio clips. Audio tag supports some file formats such as : mp3.

HTML5 supports <video> and <audio> controls. The Flash, Silverlight and similar technologies are used to play the multimedia items.

Example:

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

HTML Video Tag

HTML 5 supports <video> tag also. The HTML video tag is used for streaming video files such as a movie clip, song clip on the web page. HTML5 Video tag supports some file formats such as mp4, webM and ogg

Example:

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

HTML Tables

An HTML table is defined with the `<table>` tag. Each table row is defined with the `<tr>` tag. A table header is defined with the `<th>` tag. By default, table headings are bold and centered. A table data/cell is defined with the `<td>` tag.

Example

```
<table style="width:100%">
  <tr>    <th>Firstname</th>     <th>Lastname</th>      <th>Age</th>   </tr>
  <tr>     <td>Selam</td>    <td>Taye</td>     <td>50</td>  </tr>
  <tr>    <td>Abel</td>    <td>Mekonnen</td>     <td>44</td>  </tr>
</table>
```

Note: The `<td>` elements are the data containers of the table. They can contain all sorts of HTML elements; text, images, lists, other tables, etc.

HTML Table - Cells that Span Many Columns

To make a cell span more than one column, use the `colspan` attribute:

Example

```
<table style="width:100%">  <tr>    <th>Name</th>   <th colspan="2">Telephone</th>  </tr>
  <tr>    <td>Abel Tilahun</td>    <td>55577854</td>    <td>55577855</td>  </tr> </table>
```

HTML Table - Cells that Span Many Rows

To make a cell span more than one row, use the `rowspan` attribute:

Example

```
<table style="width:100%">  <tr>    <th>Name:</th>    <td>Abel Tilahun</td>  </tr>  <tr>    <th
rowspan="2">Telephone:</th>    <td>55577854</td>  </tr> <tr>    <td>55577855</td>  </tr></table>
```

HTML Table - Adding a Caption

To add a caption to a table, use the `<caption>` tag:

Example

```
<table style="width:100%">  <caption>Monthly savings</caption>
  <tr>    <th>Month</th>    <th>Savings</th>  </tr>  <tr>    <td>January</td>    <td>2000 birr</td>
</tr>
<tr>    <td>February</td>    <td>3000 birr</td>  </tr></table>
```

Note: The `<caption>` tag must be inserted immediately after the `<table>` tag.

HTML Lists

Unordered HTML List

An unordered list starts with the `<ul>` tag. Each list item starts with the `<li>` tag. The list items will be marked with bullets (small black circles) by default:

Example

<ul> <li>Coffee</li> <li>Tea</li> <li>Milk</li></ul>

Unordered HTML List - Choose List Item Marker

The CSS `list-style-type` property is used to define the style of the list item marker:

Value	Description
disc	Sets the list item marker to a bullet (default)
circle	Sets the list item marker to a circle
square	Sets the list item marker to a square
none	The list items will not be marked

Example - Disc

<ul style="list-style-type:disc"> <li>Coffee</li> <li>Tea</li> <li>Milk</li></ul>

Example - Circle

<ul style="list-style-type:circle"> <li>Coffee</li> <li>Tea</li> <li>Milk</li></ul>

Example - Square

<ul style="list-style-type:square"> <li>Coffee</li> <li>Tea</li> <li>Milk</li></ul>

Example - None

<ul style="list-style-type:none"> <li>Coffee</li> <li>Tea</li> <li>Milk</li></ul>

Ordered HTML List

An ordered list starts with the `<ol>` tag. Each list item starts with the `<li>` tag. The list items will be marked with numbers by default:

Example

<ol> <li>Coffee</li> <li>Tea</li> <li>Milk</li> </ol>

Ordered HTML List - The Type Attribute

The `type` attribute of the `<ol>` tag, defines the type of the list item marker:

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers

type="i"	The list items will be numbered with lowercase roman numbers

Numbers:

<ol type="1"> <li>Coffee</li> <li>Tea</li> <li>Milk</li></ol>

Uppercase Letters:

<ol type="A"> <li>Coffee</li> <li>Tea</li> <li>Milk</li></ol>

Lowercase Letters:

<ol type="a"> <li>Coffee</li> <li>Tea</li> <li>Milk</li></ol>

Uppercase Roman Numbers:

<ol type="I"> <li>Coffee</li> <li>Tea</li> <li>Milk</li></ol>

Lowercase Roman Numbers:

<ol type="i"> <li>Coffee</li> <li>Tea</li> <li>Milk</li></ol>

2.5. HTML Iframe and Forms

HTML Iframes

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

Iframe Syntax

An HTML iframe is defined with the `<iframe>` tag: <iframe src="*URL*"></iframe>

The `src` attribute specifies the URL (web address) of the inline frame page.

Iframe - Set Height and Width

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

Example

<iframe src="demo_iframe.htm" height="200" width="300"></iframe>

Or you can use CSS to set the height and width of the iframe:

Example

<iframe src="demo_iframe.htm" style="height:200px;width:300px;"></iframe>

Iframe - Remove the Border

By default, an iframe has a border around it. To remove the border, add the `style` attribute and use the CSS `border` property:

Example

<iframe src="demo_iframe.htm" style="border:none;"></iframe>

With CSS, you can also change the size, style and color of the iframe's border:

Example

<iframe src="demo_iframe.htm" style="border:2px solid red;"></iframe>

Iframe - Target for a Link

An iframe can be used as the target frame for a link.

The `target` attribute of the link must refer to the `name` attribute of the iframe:

Example

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>
<p><a href="https://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>
```

HTML Forms

The HTML `<form>` element defines a form that is used to collect user input:

<form> *form elements* </form>

An HTML form contains **form elements**. Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

Attributes of Form Tag

The Action Attribute

The `action` attribute defines the action to be performed when the form is submitted.

Normally, the form data is sent to a web page on the server when the user clicks on the submit button. In the example above, the form data is sent to a page on the server called "/action_page.html". This page contains a server-side script that handles the form data:

<form **action="/action_page.html**">

If the `action` attribute is omitted, the action is set to the current page.

The Target Attribute

The `target` attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window.

The default value is "`_self`" which means the form will be submitted in the current window.

To make the form result open in a new browser tab, use the value "`_blank`":

Example

<form action="/action_page.html" **target="_blank"**>

Other legal values are "`_parent`", "`_top`", or a name representing the name of an iframe.

The Method Attribute

The `method` attribute specifies the HTTP method (**GET** or **POST**) to be used when submitting the form data:

Example: <form action="/action_page.html" **method="get"**> or

<form action="/action_page.html" **method="post"**>

When to Use GET?

The default method when submitting form data is GET. However, when GET is used, the submitted form data will be **visible in the page address field**:

/action_page.html?firstname=Selam&lastname=Yibeltal

Notes on GET:

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user wants to bookmark the result
- GET is better for non-secure data, like query strings in Google

When to Use POST?

Always use POST if the form data contains sensitive or personal information. The POST method does not display the submitted form data in the page address field.

Notes on POST:

POST has no size limitations, and can be used to send large amounts of data.

Form submissions with POST cannot be bookmarked

The Name Attribute

Each input field must have a `name` attribute to be submitted. If the `name` attribute is omitted, the data of that input field will not be sent at all. This example will only submit the "Last name" input field:

Example

```
<form action="/action_page.html">
  First name:<br>  <input type="text"  name=”firstname” value="Selam"><br>
  Last name:<br>  <input type="text" name="lastname" value="Yibeltal"><br><br>
  <input type="submit" value="Submit">
</form>
```

HTML Form Elements

The <input> Element

The most important form element is the `<input>` element. The `<input>` element can be displayed in several ways, depending on the `type` attribute.

HTML Input types

The `<input>` element is the most important form element. The `<input>` element can be displayed in several ways, depending on the **type** attribute

Input Type Text

`<input type="text">` defines a **one-line text input field**:

Input Type Password

`<input type="password">` defines a **password field**:

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

Input Type Submit

`<input type="submit">` defines a button for **submitting** form data to a **form-handler**.

The form-handler is typically a server page with a script for processing input data. The form-handler is specified in the form's `action` attribute:

Example

```
<form action="/action_page.html">
  First name:<br>   <input type="text" name="firstname" value="Abel"><br>
  Last name:<br>   <input type="text" name="lastname" value="Degu"><br><br>
  <input type="submit" value="Submit">
</form>
```

If you omit the submit button's value attribute, the button will get a default text.

Input Type Reset

`<input type="reset">` defines a **reset button** that will reset all form values to their default values.

If you change the input values and then click the "Reset" button, the form-data will be reset to the default values.

Input Type Radio

`<input type="radio">` defines a **radio button**.

Radio buttons let a user select ONLY ONE of a limited number of choices.

Input Type Checkbox

`<input type="checkbox">` defines a **checkbox**.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Input Type Button

`<input type="button">` defines a **button**:

Example

```
<input type="button" onclick="alert('Hello World!')" value="Click Me!">
```

Input Type Date

The `<input type="date">` is used for input fields that should contain a date. Depending on browser support, a date picker can show up in the input field. You can also use the `min` and `max` attributes to add restrictions to dates:

Example

```
<form>
  Enter a date before 1980-01-01:  <input type="date" name="bday" max="1979-12-31"><br>
```

Enter a date after 2000-01-01: <input type="date" name="bday" min="2000-01-02">
</form>

Input Type Email

The `<input type="email">` is used for input fields that should contain an e-mail address. Depending on browser support, the e-mail address can be automatically validated when submitted. Some smartphones recognize the email type, and adds ".com" to the keyboard to match email input.

Input Type File

The `<input type="file">` defines a file-select field and a "Browse" button for file uploads.

Input Type Number

The `<input type="number"` min="1" max="5">> defines a **numeric** input field. You can also set restrictions on what numbers are accepted.

Input Restrictions

Here is a list of some common input restrictions (some are new in HTML5):

Attribute	Description
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

Input Type Range

The `<input type="range">` defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the `min`, `max`, and `step` attributes:

Example

<form><input type="range" name="points" min="0" max="10"></form>

Input Type Search

The `<input type="search">` is used for search fields (a search field behaves like a regular text field).

Input Type Time

The `<input type="time">` allows the user to select a time (no time zone). Depending on browser support, a time picker can show up in the input field.

The <select> Element

The `<select>` element defines a **drop-down list**:

Example

```
<select name="cars">  <option value="volvo">Volvo</option>
                      <option value="saab">Saab</option>
                      <option value="fiat">Fiat</option>
                      <option value="audi">Audi</option>
</select>
```

The `<option>` elements defines an option that can be selected. By default, the first item in the drop-down list is selected. To define a pre-selected option, add the `selected` attribute to the option:

Example

```
<option value="fiat" selected>Fiat</option>
```

Visible Values:

Use the `size` attribute to specify the number of visible values:

Example

```
<select name="cars" size="3">  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option></select>
```

Allow Multiple Selections:

Use the `multiple` attribute to allow the user to select more than one value:

Example

```
<select name="cars" size="4" multiple>  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option></select>
```

The <textarea> Element

The `<textarea>` element defines a multi-line input field (**a text area**):

Example

<textarea name="message" rows="10" cols="30" style="width:200px; height:600px"> The cat was playing in the garden. </textarea>

The `rows` attribute specifies the visible number of lines in a text area. The `cols` attribute specifies the visible width of a text area.

Grouping Form Data with <fieldset>

The `<fieldset>` element is used to group related data in a form. The `<legend>` element defines a caption for the `<fieldset>` element.

Example

```
<form action="/action_page.html">  <fieldset>
   <legend>Personal information:</legend>
   First name:<br>    <input type="text" name="firstname" value="Abel"><br>
   Last name:<br>    <input type="text" name="lastname" value="Degu"><br><br>
   <input type="submit" value="Submit">
 </fieldset></form>
```

The form Attribute

The `form` attribute specifies one or more forms an `<input>` element belongs to.

- ✓ The `multiple` attribute specifies that the user is allowed to enter more than one value in the `<input>` element. The `multiple` attribute works with the following input types: email, and file. Example: Select images: <input type="file" name="img" multiple>
- ✓ The `pattern` attribute specifies a regular expression that the `<input>` element's value is checked against. The `pattern` attribute works with the following input types: text, search, url, tel, email, and password. Example: <input type="text" name="country_code" pattern="[A-Za-z]{3}" title="Three letter country code">
- ✓ The `placeholder` attribute specifies a hint that describes the expected value of an input field (a sample value or a short description of the format). The hint is displayed in the input field before the user enters a value. The `placeholder` attribute works with the following input types: text, search, url, tel, email, and password.
- ✓ The `required` attribute specifies that an input field must be filled out before submitting the form. The `required` attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

Migration from HTML4 to HTML5

This section demonstrates how to convert an HTML4 page into an HTML5 page, without destroying anything of the original content or structure.

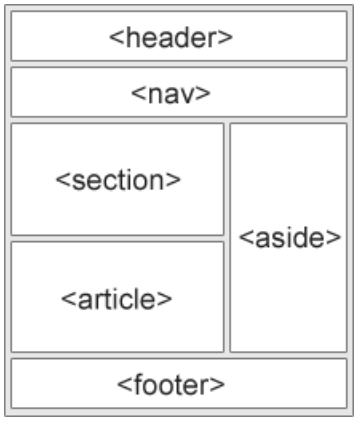
Typical HTML4	Typical HTML5
	<header>
	<nav>
	<section>
	<article>
	<footer>

HTML Layouts

HTML Layout Elements

Websites often display content in multiple columns (like a magazine or newspaper).

HTML5 offers new semantic elements that define the different parts of a web page:



- <header> - Defines a header for a document or a section
- <nav> - Defines a container for navigation links
- <section> - Defines a section in a document
- <article> - Defines an independent self-contained article
- <aside> - Defines content aside from the content (like a sidebar)
- <footer> - Defines a footer for a document or a section
- <details> - Defines additional details
- <summary> - Defines a heading for the <details> element

HTML Layout Techniques

There are five different ways to create multicolumn layouts. Each way has its pros and cons:

- HTML tables (not recommended)
- CSS float property
- CSS flexbox
- CSS framework
- CSS grid

HTML Tables

The <table> element was not designed to be a layout tool! The purpose of the <table> element is to display tabular data. So, do not use tables for your page layout! They will bring a mess into your code. And imagine how hard it will be to redesign your site after a couple of months.

Tip: Do NOT use tables for your page layout!

CSS Frameworks

If you want to create your layout fast, you can use a framework, like W3.CSS or Bootstrap.

CSS Floats

It is common to do entire web layouts using the CSS float property. Float is easy to learn - you just need to remember how the float and clear properties work.

Disadvantages: Floating elements are tied to the document flow, which may harm the flexibility.

CSS Flexbox

Flexbox is a new layout mode in CSS3. Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

Disadvantages: Does not work in IE10 and earlier.

CSS Grid View

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

Disadvantages: Does not work in IE nor in Edge 15 and earlier.

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

you cannot use double slashes (//) for comments in CSS. /* kana qofa fayadamna*/

Chapter 3

Cascading style sheets (CSS)

Introduction to CSS

CSS stands for **Cascading Style Sheet**, and it allows us to separate our web sites HTML content from its style. As always we use HTML file to arrange the content, but all of the presentation/formatting (fonts, colors, background, borders, text formatting, link effects & so on...) are accomplished within a CSS.

CSS is a web page layout method that has been added to HTML to give web developers more control over their design and content layout. Using **CSS** allows a designer to **create** a standard set of commands (either embedded inside the web page or from an external page) that controls the style of all subsequent pages.

CSS separates the layout and the styles of a web page. This is often difficult to comprehend for web designers that are used to compiling their creative and HTML coding in a single web page document. Styles such as fonts, font sizes, margins, can be specified in one place, and then the Web pages feed off this one master list, with the styles cascading throughout the page or an entire site.

It is more and more difficult to create Web sites where the content of HTML documents was clearly separated from the document's presentation layout. The layout of the document was supposed to be taken care of by the browser, without using any formatting tags. This is because styles sheets define **HOW HTML** elements are to be displayed, just like the font tag and the color attribute in HTML.

CSS is a breakthrough in web design because it allows developers to control the **style** and **layout** of multiple web pages all at once. As a web developer, it is possible to define a style for each HTML element and apply it to as many web pages as we want. To make a global change, simply change the style, and all elements in the Web are updated automatically.

Style sheets allow style information to be specified in many ways. Styles can be specified **inside** a single HTML element, inside the **<head>** element of an HTML page, or in an external CSS file. Even multiple external style sheets can be referenced inside a single HTML document.

What is CSS?

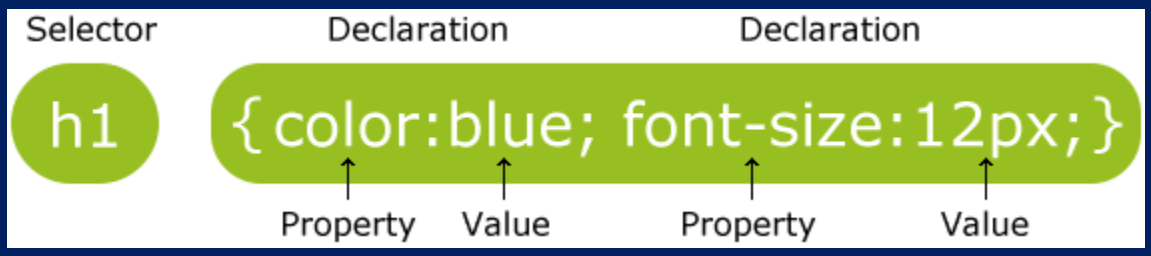
- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**

CSS Saves a Lot of Work!

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

CSS Syntax

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



- The selector is normally the **HTML** element you want to **style**.
- Each declaration consists of a **property** and a **value**.
- The property is the style **attribute** you want to change. Each property has a **value**.
- CSS declarations always ends with a **semicolon**, and declaration groups are surrounded by **curly brackets**:

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

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

Example

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

CSS Comments

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

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

Grouping Content

The ** ** and **** elements allow you to group together several elements to create sections or subsections of a page.

This is the very important block level tag which plays a big role in grouping various other HTML tags and applying CSS on group of elements. Even now tag can be used to create webpage layout where we define different parts (Left, Right, Top etc.) of the page using tag. This tag does not provide any visual change on the block but this has more meaning when it is used with CSS.

For example, you might want to put all of the footnotes on a page within a ** ** element to indicate that all of the elements within that ** ** element relate to the footnotes. You might then attach a style to this ** ** element so that they appear using a special set of **style rules**.

Example 1:

```
<!DOCTYPE html>
<html>
<head>
<title> Div Tag Example</title>
</head>
<body>
<div id="menu" align="middle" >
<a href="/index.htm">HOME</a> |
<a href="/about/contact_us.htm">CONTACT</a> |
<a href="/about/index.htm">ABOUT</a>
</div>
<div id="content" align="left" bgcolor="white">
<h5>Content Articles</h5>
<p>Actual content goes here.....</p>
</div>
</body>
```

```
</html>
```

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML div Tag</title>
</head>
<body>

<!-- First group of tags -->
<div style="color:red">
<h4>This is first group</h4>
<p>Following is a list of vegetables</p>
<ul>
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ul>
</div>

<!-- Second group of tags -->
<div style="color:green">
<h4>This is second group</h4>
<p>Following is a list of fruits</p>
<ul>
<li>Apple</li>
<li>Banana</li>
<li>Mango</li>
<li>Strawberry</li>
</ul>
</div>
</body>
</html>
```

The **** element, on the other hand, can be used to group inline elements only. So, if you have a part of a sentence or paragraph which you want to group together, you could use the **** element as follows.

The HTML **** is an inline element and it can be used to group inline-elements in an HTML document. This tag also does not provide any visual change on the block but has more meaning when it is used with CSS.

Example1

```
<!DOCTYPE html>
<html>
<head>
<title>Span Tag Example</title>
</head>
<body>
```

```
<p>This is the example of <span style="color:green">span tag</span> and the <span style="color:red">div
tag</span> alongwith CSS</p>
</body>
</html>
```

Example 2

```
<!DOCTYPE html>
<html>
<head>
<title>HTML span Tag</title>
</head>
<body>
<p>This is <span style="color:red">red</span> and this is <span
style="color:green">green</span></p>
</body>  </html>
```

- The difference between the **** tag and the ** ** tag is that the tag is used with inline elements whereas the tag is used with block-level elements.

CSS Id and Class Selector

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

The id Selector

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

Example

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

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

The class Selector

- The class selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements.
- This allows you to set a particular style for any HTML elements with the **same** class.
- The **class** selector uses the HTML class attribute, and is defined with a "**.**"

In the example below, all HTML elements with class="center" will be center-aligned:

Example

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

- You can also specify that only specific HTML elements should be affected by a class.

In the example below, all **p** elements with class="center" will be **center-aligned**:

Example

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

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

CSS How To

When a browser reads a style sheet, it will format the document according to it.

Three Ways to Insert CSS

There are **three** ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

- **External Style Sheet**: Define style sheet rules in a separate .css file and then include that file in your HTML document using HTML **<link>** tag.
- **Internal Style Sheet**: Define style sheet rules in header section of the HTML document using **<style>** tag.
- **Inline Style Sheet**: Define style sheet rules directly along-with the HTML elements using **style** attribute.

External Style Sheet

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

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

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

```
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
```

Save **sytle.css**

```
/* CSS class */
.red{
color: Blue;
}
.thick{
font-size:20px;
}
.green{
color:green;
}
```

```
<!DOCTYPE html>
<html>
<head>
<title>HTML External CSS</title>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<p class="red">This is red</p>
<p class="thick">This is thick</p>
<p class="green">This is green</p>
```

```
<p class="thick green">This is thick and green</p>

</body>

</html>
```

An internal style sheet should be used when a single document has a unique style.

- Do not leave spaces between the property value and the units! "margin-left:20 px" (instead of "margin-left:20px") will work in **IE**, but **not** in **Firefox or Opera**.

Internal Style Sheet

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

```
<head>
<style type="text/css">
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Internal CSS</title>
<style type="text/css">
.red{
color: red;
}
.thick{
font-size:20px;
}
.green{
color:green;
}
</style>
</head>
<body>
<p class="red">This is red</p>
<p class="thick">This is thick</p>
<p class="green">This is green</p>
<p class="thick green">This is thick and green</p>
</body>
```

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

sparingly!=in a restricted or infrequent manner; in small quantities.

```
</html>
```

Inline Styles

An **inline style** loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly!
To use inline styles you use the **style** attribute in the relevant tag. The style attribute can contain any **CSS** property. The example shows how to change the **color** and the **left margin** of a paragraph:

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

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Inline CSS</title>
</head>
<body>
<p style="color:red;">This is red</p>
<p style="font-size:20px;">This is thick</p>
<p style="color:green;">This is green</p>
<p style="color:green;font-size:20px;">This is thick and green</p>
</body>
</html>
```

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet

Multiple Style Sheets

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

```
h3
{
color:red;
text-align:left;
font-size:8pt;
}
```

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

```
h3
{
text-align:right;
font-size:20pt;
}
```

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

```
color:red;
text-align:right;
font-size:20pt;
```

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

Multiple Styles Will Cascade into One

Styles can be specified:

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

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

Cascading order

What style will be used when there is more than **one style** specified for an **HTML** element?

Generally speaking we can say that all the styles will **"cascade"** into a new "**virtual**" style sheet by the following rules, where number four has the **highest** priority:

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

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

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

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

CSS Background

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

CSS properties used for background effects:

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

Background Color

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

Example

```
body {background-color:#b0c4de;}
```

The background color can be specified by:

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

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

Example

```
h1 {background-color:#6495ed;}
p {background-color:#e0ffff;}
div {background-color:#b0c4de;}
```

Background Image

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

Example

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

Below is an example of a bad combination of text and background image. The text is almost not readable:

Example

```
body {background-image:url('bgdesert.jpg');}
```

Background Image - Repeat Horizontally or Vertically

By default, the **background-image** property repeats an image both **horizontally** and **vertically**. Some images should be repeated only horizontally or vertically, or they will look strange, like this:

Example

```
body
{

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

If the image is repeated only horizontally (repeat-x), the background will look better:

Example

```
body
{
background-image:url('gradient2.png');
background-repeat:repeat-x;
}
```

Background Image - Set position and no-repeat

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

Example

```
body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;
}
```

In the example above, the background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the **text** too much.
The position of the image is specified by the **background-position** property:

Example

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

All CSS Background Properties

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

Property	Description	Values	CSS
background	Sets all the background properties in one declaration	*background-color background-image background-repeat background-*	1

		attachment background-position inherit	
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page	scroll fixed inherit	1
background-color	Sets the background color of an element	*color-rgb* *color-hex* *color-name* transparent inherit	1
background-image	Sets the background image for an element	url(*URL*) none inherit	1
background-position	Sets the starting position of a background image	left top left center left bottom right top right center right bottom center top center center center bottom *x% y%* *xpos ypos* inherit	1
background-repeat	Sets if/how a background image will be repeated	repeat repeat-x repeat-y no-repeat inherit	1

CSS Text

Text formatting

This text is styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties. The paragraph is indented, aligned, and the space between characters is specified. The underline is removed from the "Try it yourself" link.

Text Color

The color property is used to set the color of the text. The color can be specified by:

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

The default color for a page is defined in the body selector.

Example

```
body {color:blue;}
h1 {color:#00ff00;}
h2 {color:rgb(255,0,0);}
```

Text Alignment

The text-align property is used to set the horizontal alignment of a text.

Text can be centred , or aligned to the left or right, or justified.

When text-align is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

Example

```
h1 {text-align:center;}
p.date {text-align:right;}
p.main {text-align:justify;}
```

Text Decoration

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

Example

```
a {text-decoration:none;}
```

It can also be used to decorate text:

Example

```
h1 {text-decoration:overline;}
h2 {text-decoration:line-through;}
h3 {text-decoration:underline;}
h4 {text-decoration:blink;}
```

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

The text-transform property is used to specify uppercase and lowercase letters in a text

Text Transformation

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

Example

```
p.uppercase {text-transform:uppercase;}
p.lowercase {text-transform:lowercase;}
p.capitalize {text-transform:capitalize;}
```

Text Indentation

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

Example

```
p {text-indent:50px;}
```

All CSS Text Properties

The number in the "**CSS**" column indicates in which **CSS** version the property is defined (**CSS1** or **CSmS2**).

Property	Description	Values	CSS
color	Sets the color of a text	*color*	1
direction	Sets the text direction	ltr rtl	2
line-height	Sets the distance between lines	normal *number* *length* *%*	1
letter-spacing	Increase or decrease the space between characters	normal *length*	1
text-align	Aligns the text in an element	left right center justify	1
text-decoration	Adds decoration to text	none underline overline line-through blink	1
text-indent	Indents the first line of text in an element	*length* *%*	1
text-shadow		none *color* *length*	
text-transform	Controls the letters in an element	none capitalize uppercase lowercase	1
unicode-bidi		normal embed bidi-override	2
vertical-align	Sets the vertical alignment of an element	baseline sub super top text-top middle bottom	1

		text-bottom *length* *%*	
white-space	Sets how white space inside an element is handled	normal pre nowrap	1
word-spacing	Increase or decrease the space between words	normal *length*	1

CSS Font

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

Difference Between Serif and Sans-serif Fonts

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

CSS Font Families

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

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

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

---

Font Family

The font family of a text is set with the font-family property.

The font-family property should hold several font names as a "**fallback**" system. If the browser does not support the first font, it **tries** the next font.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

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

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

Example

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

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

Font Style

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

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

Example

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

Font Size

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

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

Relative size:

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

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

Set Font Size With Pixels

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

Example

```
h1 {font-size:40px;}
h2 {font-size:30px;}
p {font-size:14px;}
```

The example above allows Firefox, Chrome, and Safari to resize the text, **but not Internet Explorer**.
The text can be resized in all browsers using the zoom tool (however, this resizes the entire page, not just the text).

Set Font Size With Em

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

Example

```
h1 {font-size:2.5em;} /* 40px/16=2.5em */
h2 {font-size:1.875em;} /* 30px/16=1.875em */
p {font-size:0.875em;} /* 14px/16=0.875em */
```

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.
Unfortunately, there is still a problem with IE. When resizing the text, it becomes larger than it should when made larger, and smaller than it should when made smaller.

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the body element:

Example

```
body {font-size:100%;}
h1 {font-size:2.5em;}
h2 {font-size:1.875em;}
p {font-size:0.875em;}
```

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

All CSS Font Properties

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

Property	Description	Values	CSS
font	Sets all the font properties in one declaration	*font-style* *font-variant* *font-weight* *font-size/line-height* *font-family* caption icon menu message-box small-caption status-bar inherit	1
font-family	Specifies the font family for text	*family-name* *generic-family* inherit	1
font-size	Specifies the font size of text	xx-small x-small small medium large x-large xx-large smaller larger *length* *%* inherit	1
font-style	Specifies the font style for text	normal italic oblique inherit	1
font-variant	Specifies whether or not a text should be displayed in a small-caps font	normal small-caps inherit	1
font-weight	Specifies the weight of a font	normal bold bolder lighter 100 200 300 400 500 600 700 800 900 inherit	

CSS Links

Links can be styled in different ways.

Styling Links

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

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

Example

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

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

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

Common Link Styles

In the example above the link changes color depending on what state it is in.
Lets go through some of the other common ways to style links:

Text Decoration

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

Example

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}
```

Background Color

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

Example

```
a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

CSS Lists

The CSS list properties allow you to:

- Set different list item markers for **ordered** lists
- Set different list item markers for **unordered** lists

- Set an image as the list item marker

List

In HTML, there are two types of lists:

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

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

Different List Item Markers

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

Example

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

ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
```

Some of the property values are for unordered lists, and some for ordered lists.

Values for Unordered Lists

Value	Description
None	No marker
Disc	Default. The marker is a filled circle
Circle	The marker is a circle
Square	The marker is a square

Values for Ordered Lists

Value	Description
Armenian	The marker is traditional Armenian numbering
Decimal	The marker is a number
decimal-leading-zero	The marker is a number padded by initial zeros (01, 02, 03, etc.)
Georgian	The marker is traditional Georgian numbering (an, ban, gan, etc.)
lower-alpha	The marker is lower-alpha (a, b, c, d, e, etc.)
lower-greek	The marker is lower-greek (alpha, beta, gamma, etc.)
lower-latin	The marker is lower-latin (a, b, c, d, e, etc.)
lower-roman	The marker is lower-roman (i, ii, iii, iv, v, etc.)

upper-alpha	The marker is upper-alpha (A, B, C, D, E, etc.)
upper-latin	The marker is upper-latin (A, B, C, D, E, etc.)
upper-roman	The marker is upper-roman (I, II, III, IV, V, etc.)

Note: No versions of Internet Explorer (including IE8) support the property values "decimal-leading-zero", "lower-greek", "lower-latin", "upper-latin", "armenian", or "georgian" UNLESS a DOCTYPE is specified!

An Image as The List Item Marker

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

Example

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

The example above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari.

If you want the image-marker to be placed equally in all browsers, a crossbrowser solution is explained below.

Crossbrowser Solution

The following example displays the image-marker equally in all browsers:

Example

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

Example explained:

- For ul:
  - Set the list-style-type to none to remove the list item marker
  - Set both padding and margin to 0px (for cross-browser compatibility)

- For li:
  - Set the URL of the image, and show it only once (no-repeat)
  - Position the image where you want it (left 0px and down 5px)
  - Position the text in the list with padding-left

All CSS List Properties

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

Property	Description	Values	CSS
list-style	Sets all the properties for a list in one declaration	*list-style-type* *list-style-position* *list-style-image* inherit	1
list-style-image	Specifies an image as the list-item marker	URL none inherit	1
list-style-position	Specifies if the list-item markers should appear inside or outside the content flow	inside outside inherit	1
list-style-type	Specifies the type of list-item marker	none disc circle square decimal decimal-leading-zero armenian georgian lower-alpha upper-alpha lower-greek lower-latin upper-latin lower-roman upper-roman inherit	

CSS Tables

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

Table Borders

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

Example

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

```
}
```

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

Collapse Borders

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

Example

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

Table Width and Height

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

Example

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

Table Text Alignment

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

Example

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

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

Example

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

Table Padding

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

Example

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

Table Color

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

Example

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

HTML – LAYOUTS

A webpage layout is very important to give better look to your website. It takes considerable time to design a website's layout with great look and feel.

Now- a-days, all modern websites are using CSS and JavaScript based framework to come up with responsive and dynamic websites but you can create a good layout using simple HTML tables or division tags in combination with other formatting tags. This chapter will give you few examples on how to create a simple but working layout for your webpage using pure HTML and its attributes.

HTML Layout - Using Tables

The simplest and most popular way of creating layouts is using HTML <table> tag. These tables are arranged in columns and rows, so you can utilize these rows and columns in whatever way you like.

Example

For example, the following HTML layout example is achieved using a table with 3 rows and 2 columns but the header and footer column spans both columns using the colspan attribute:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Layout using Tables</title>
</head>
<body>
<table width="100%" border="0">
<tr>
<td colspan="2" bgcolor="#b5dcb3">
<h1>This is Web Page Main title</h1>
</td>
</tr>
<tr valign="top">
<td bgcolor="#aaa" width="50">
<b>Main Menu</b><br />
HTML<br />
PHP<br />
PERL...
</td>
<td bgcolor="#eee" width="100" height="200">
Technical and Managerial Tutorials
</td>
</tr>
<tr>
```

```
<td colspan="2" bgcolor="#b5dcb3">
<center>
Copyright © 2007 Tutorialspoint.com
</center>
</td>
</tr>
</table>
</body>
</html>
```

This will produce the following result:

Multiple Columns Layout - Using Tables

You can design your webpage to put your web content in multiple pages. You can keep your content in middle column and you can use left column to use menu and right column can be used to put advertisement or some other stuff. This layout will be very similar to what we have at our website tutorialspoint.com.

```
<!DOCTYPE html>
<html>
<head>
<title>Three Column HTML Layout</title>
</head>
<body>
```

```
<table width="100%" border="0">
<tr valign="top">
<td bgcolor="#aaa" width="20%">
<b>Main Menu</b><br />
HTML<br />
PHP<br />
PERL...
</td>
<td bgcolor="#b5dcb3" height="200" width="60%">
Technical and Managerial Tutorials
</td>
<td bgcolor="#aaa" width="20%">
<b>Right Menu</b><br />
HTML<br />
PHP<br />
PERL...
</td>
</tr>
<table>
</body>
</html>
```

This will produce the following result:

Main Menu HTML PHP PERL...	Technical and Managerial Tutorials	**Right Menu** HTML PHP PERL...

HTML Layouts - Using DIV, SPAN

The element is a block level element used for grouping HTML elements. While the tag is a block-level element, the HTML element is used for grouping elements at an inline level.

Although we can achieve pretty nice layouts with HTML tables, but tables weren't really designed as a layout tool. Tables are more suited to presenting tabular data.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Layouts using DIV, SPAN</title>
</head>
```

```
<body>
<div style="width:100%">
<div style="background-color:#b5dcb3; width:100%">
<h1>This is Web Page Main title</h1>
</div>
<div style="background-color:#aaa; height:200px;width:100px;float:left;">
<div><b>Main Menu</b></div>
HTML<br />
PHP<br />
PERL...
</div>
<div style="background-color:#eee; height:200px;width:350px;float:left;">
<p>Technical and Managerial Tutorials</p>
</div>
<div style="background-color:#aaa; height:200px;width:100px;float:right;">
<div><b>Right Menu</b></div>
HTML<br />
PHP<br />
PERL...
</div>
<div style="background-color:#b5dcb3;clear:both">
<center>
Copyright © 2007 Tutorialspoint.com
</center>
</div>
</div>
</body>
</html>
```

Chapter 4

JavaScript Programming

4.1. Introduction

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more. JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, and Opera.

Scripting languages are usually interpreted rather than compiled. That means that a software routine called an interpreter translate a program's statements into machine code, code understandable by a particular type of computer, before executing them every time the program is run. Compiled languages, on the other hand, are translated into machine code and stored for later execution. When the compiled program is run, it executes immediately without further need of interpretation; it was interpreted into machine code when it was compiled. Because programs written in interpreted languages must be translated into machine code every time they are run, they are typically slower than compiled programs. However, this does not usually present a problem for the small applications for which scripting languages are generally used.

Being interpreted does have its advantages. One is platform independence. Because an interpreter performs the translation, we can write the program once and run it on a variety of platforms. In the case of JavaScript, the interpreter is built into Web browsers. Browsers are available for a variety of platforms and operating systems. Another advantage is that scripting languages are often loosely typed and more forgiving than compiled languages.

4.2.Purposes of Java script

- **JavaScript gives HTML designers a programming tool -** JavaScript is a scripting language with a very simple syntax!
- **JavaScript can put dynamic text into an HTML page -** A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page
- **JavaScript can react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from **extra** processing.

- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

4.3. Running the JavaScript

Any time including JavaScript in an HTML document, we must enclose those lines inside a <**SCRIPT**>...</**SCRIPT**> tag pair. These tags alert the browser program to begin interpreting all the text between these tags as a script. Because other scripting languages (such as Microsoft's VBScript) can take advantage of these script tags, we must specify the precise name of the language in which the enclosed code is written. Therefore, when the browser receives this signal that our script uses the **JavaScript** language, it employs its built-in JavaScript interpreter to handle the code.

JavaScript is case-sensitive. Hence, we must enter any item in our script that uses a JavaScript word with the correct uppercase and lowercase letters. HTML tags (including the <SCRIPT> tag) can be in the case of our choice, but everything in JavaScript is case-sensitive. JavaScript should be pt between the following:-

```
<SCRIPT LANGUAGE="text/JavaScript">
 //your script here
</SCRIPT>
```

Or

```
<SCRIPT type="text/JavaScript">
 //your script here
</SCRIPT>
```

As shown above, the attribute name called LANGUAGE & TYPE can be used interchangeably. But, commonly we use type attribute.

Here are some tips to remember when writing JavaScript commands:

- JavaScript code is case sensitive (e.g. age and Age are different variables)
- White space between words and tabs are ignored
- Line breaks are ignored except within a statement
- JavaScript statements end with a semi colon (;) but not always

Javascript can come at different places. There is a flexibility given to include JavaScript code anywhere in an HTML document. But there are following most preferred ways to include JavaScript in an HTML file.

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in and **external file** and then include in <head>...</head> section.

❖ Wherever we use/ define a JavaScript block in our web page, simply use the following block of HTML.

<script type='text/JavaScript'>

// Your script goes here.

</script>

❖ We can place these script blocks anywhere on the page that we wish, there are some rules and conventions however.

❖ But, for external e**xternal File,** Use the **SRC** attribute of the <SCRIPT> tag to call JavaScript code from an **external text** file.

❖ This is useful if we have a lot of code or we want to run it from several pages, because any number of pages can call the same external JavaScript file. The text file itself contains no HTML tags. It is called by the following tag:

<**SCRIPT type** = "JavaScript" SRC = *" **url of javascript file"***> </SCRIPT>

❖ If we want to run the same JavaScript on several pages, without having to write the same script on every page, we can write a JavaScript in an **external file**. Save the external JavaScript file with **a .js** file extension. The external script cannot contain the <script> </script> tags. To use the external script, point to the **.js** file in the "**src**" attribute of the <script> tag.

❖ JavaScripts is put in a page where it will be executed immediately while the page loads into the browser. Sometimes we want to execute a script when a page loads, or at a later event, such as when a user clicks a button.

Script between <head>…..</head>

Scripts to be executed when they are called, or when an event is triggered, are placed in functions. Put functions in the head section, this way they are all in one place, and they do not interfere with page content.

```
<html>
<head>
<script language="javascript">
function message()
{
alert("This alert box was called with the onload event");
}
</script>
</head>
<body onload="message()">
</body>
</html>
```

Scripts in <body>…..</head> tags

If we don't want our script to be placed inside a function, or if our script should write page content, it should be placed in the body section.

```
<html>
<head>  </head>
<body>
<script language="javascript">
document.write ("This message is written by JavaScript");
</script>
</body>
</html>
```

Scripts in both <head>…..</head> and <body>…..</body> tags

We can place an unlimited number of scripts in our document, so we can have scripts in both the body and the head section.

4.4.Input-Output in Java/ Message Boxes

In JavaScript, input-output can be done in different ways:

- ***documents.write*** method writes a string to the web page. Anything between double quotes will be displayed as it is in the web page. However, if there is something out of quotes, it is evaluated as expression and the result will be sent to the web page.
- The ***alert*** method is produces a browser alert box. These are incredibly useful for debugging and learning the language. However, they are not good way to communicate with the users. An alert box is often used if we want to make sure information comes through to the user. When an alert **box pops up**, the user will have to click "OK" to proceed.

- The **Syntax is**

  window.alert("*sometext*"); or simply write as **alert("some text")**

  Example belows:-

```
<html>
<head>
<script language="javascript">
document.write("This message is written by JavaScript");
function message()
{
alert("message to display");
}
</script>
 </head>
<body onload="message()">
<br><br><br><br><br><br>
<h1>I THANK YOU FOR EVERY THING YOU MADE!!!</H1>
</body>
</html>
```

An alert box is often used if we want to make surei

- **Prompt box:-** is used to allow a user to enter something according to the promotion. It is written as **window.prompt**("please enter your name") or simply write as prompt("some text")

  Example:-

```
<script>
var y=window.prompt("please enter your name")
window.alert(y)
</script>
```

- **Confirm Box:-** A confirm box is often used if we want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false. It has the following Syntax window.confirm("*sometext*"); or simply write confirm("some text")

  Example :-

```
<script>
var x=window.confirm("Are you sure you want to quit")
if (x)
   window.alert("Thank you.")
     x="You pressed OK!";
else
```

```
    window.alert("Good choice.")
     x="You pressed Cancel!";
document.write (x);
</script>
```

- **The Page Printing:-** JavaScript helps us to implement this functionality using **print** function of *window* object. The JavaScript print function **window.print()** will print the current web page when executed. We can call this function directly using ***onclick*** event as follows:

```
<html>
<head>
<script type="text/javascript">
</script>
</head>
<body>
<form>
<input type="button" value="Print" onclick="window.print()" />
</form>
</body>
```

wasagni dha window,print() call godhuun pag se ana pring gochuunn ni danda'ama

4.5.Working with Variables and Data

In JavaScript, a value can be one of several types as shown bellow:-

Type	Example	Description
String	"John"	a series of characters inside quotation marks
Number	4.5	any number not inside quotes
Boolean	True	a logical true or false
Null	Null	completely devoid of any value
Object		a software thing that is defined by its properties and methods

Table JavaScript data types

i. Declaring Variables

To declare variable, we use the **var** keyword, followed by the name we want to give to the **variable**. JavaScript variables can be declared with the *var* keyword as

var ***variablename;***

To declare a new variable called **myAge**, the JavaScript statement is var myAge;

- Variable names obey **number** of restrictions (rules).
  - ✓ Don't use any reserved keyword as a variable name.
  - ✓ A variable name cannot contain space characters. Therefore, one-word variable names are fine. For example, the followings are invalid:- var my_age; var myAge;
  - ✓ Variable names should avoid all punctuation symbols except for the underscore character. Also, the first character of a variable name **cannot** be a numeral.
  - ✓ To put together, a variable name can contain letters of the **alphabet**, both capital and small, number and _(underscore). Still, the name should start with letters of the alphabet, or _, not digits

Example**:** see the following correct variable declaration,

```
var firstName;
var he89;
var TH_;
```

Example**:** Wrong variable names:

```
Var first name; //space not allowed
Var 89he; //can't start with digit
Var TH.; //punctuation mark not allowed
```

ii. Assigning Values to Variables

After the declaration shown above, the variables are empty which means they have no data **values** yet. After declaring a variable, it is possible to assign a value to it by using equal **sign** (=) as follows:

```
var myAge;
myAge = 25;
```

However, we can also assign values to the variables as:

```
var age=5;
```

iii. Assigning Values to Undeclared Variables

To assign values to variables that have not yet been declared, the variables will automatically be declared. These statements:

```
age=5;
```

The above statements have the same effect as:

```
var age=5;
```

iv. Redeclaring JavaScript Variables

If we redeclare a JavaScript variable, it will not lose its original value.

```
var age=5;
var age;
```

After the execution of the statements above, the variable age will still have the value of 5. The value of age is not reset (or cleared) when we redeclare it.

- Comments are ignored by the interpreter. The purpose of including comments is to remind us how our script works after long period has passed when we see our code.

- JavaScript supports two types of comments:
  - Comments on a single line are preceded by a double-slash (//).
  - Comments that span multiple lines are preceded by /* and followed by */

4.6.Operators and Expressions

An operator performs some kind of calculation (operation) or comparison with two values to reach a third value. Generally, operators can be broadly categorized into two: mathematical operators and comparison and logical operators.

i. Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations between variables or values. +, -, *, /, %, ++, --

ii. JavaScript Assignment Operators

Assignment operators are used to perform arithmetic operation and then assign the result to variables. Includes:- =, +=, -=, *=, /= and %=

iii. Comparison Operators

Comparison operators help to compare two or more values — whether two values are equal, for example. These kinds of comparisons return a value of the Boolean type true or false.

- Includes: - >, >=, <, <=, ==, and! =

iv. Logical Operators

Logical operators are used to determine the logic between variables or values:- &&, ||

V. The Bitwise Operators:

There are following bitwise operators supported by JavaScript language Assume variable A holds 2 and variable B holds 3 then:

Operator	Description	Example
&	Called Bitwise AND operator. It performs a Boolean AND operation on each bit of its integer arguments.	(A & B) is 2 .
|	Called Bitwise OR Operator. It performs a Boolean OR operation on each bit of its integer arguments.	(A | B) is 3.
^	Called Bitwise XOR Operator. It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both.	(A ^ B) is 1.
~	Called Bitwise NOT Operator. It is a is a unary operator and operates by reversing all bits in the operand.	(~B) is -4 .
<<	Called Bitwise Shift Left Operator. It moves all bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying by 2, shifting two positions is equivalent to multiplying by 4, etc.	(A << 1) is 4.
>>	Called Bitwise Shift Right with Sign Operator. It moves all bits in its first operand to the right by the number of places specified in the second operand. The bits filled in on the left depend on the sign bit of the original operand, in order to preserve the sign of the result. If the first operand is positive, the result has zeros placed in the high bits; if the first operand is negative, the result has ones placed in the high bits. Shifting a value right one place is equivalent to dividing by 2 (discarding the remainder), shifting right two places is equivalent to integer division by 4, and so on.	(A >> 1) is 1.
>>>	Called Bitwise Shift Right with Zero Operator. This operator is just like the >> operator, except that the bits shifted in on the left are always zero,	(A >>> 1) is 1.

4.7.Java Scrpit Statements

A. Working with Conditional Statements

Conditional statements are used to perform different actions based on different conditions. Broadly, there are two ways to execute code conditionally:

- If statement
- switch statement

i. If condition/if -----else condition:- The simplest program decision is to follow a special branch or path of the program if a certain condition is true. Formal **syntax** for this construction follows:

if (*condition*) {
statement[s] if true
}

The keyword ***if*** is a must. In the parentheses goes an expression that evaluates to a Boolean value. This is the condition being tested as the program runs past this point. If the condition evaluates to **true**, then one or more statements inside the curly braces execute before continuing on with the next statement after the **closing** brace. If the condition evaluates to **false**, then the statements inside the curly brace are ignored and processing continues with the next statement after the closing brace. Items in italics are replaced with expressions and statements that fit the situation.

The formal syntax definition for an **if...else** construction is as follows:

if (*condition*)
{
statement(s) if true
}
[else if(*condition*){
statement(s) if true
}]
[else {
statement(s) if false
}]

Where [] indicates optional parts of the **JavaScript** code.

Example1:-

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

Example:-

```
<script type="text/javascript">
var book = "maths";
if( book == "history" ){
  document.write("<b>History Book</b>");
}else if( book == "maths" ){
  document.write("<b>Maths Book</b>");
}else if( book == "economics" ){
  document.write("<b>Economics Book</b>");
}else{
 document.write("<b>Unknown Book</b>");
}
</script>
```

ii. Switch Statement:- A switch statement allows a program to evaluate an **expression** and attempt to match the expression's value to a **case label.** If a match is found, the program executes the **associated** statement. A switch statement looks as follows:

```
Switch (expression)
{
  case label1:
    statements1
    [break;]
  case label2:
    statements2
    [break;]
  ...
```

```
default:
  statementsdefault
  [break;]}
```

Example 1:-

```
<script type="text/javascript">
var d=new Date();
theDay=d.getDay();
switch (theDay)
{ case 5: document.write("Finally Friday");
break;
case 6: document.write("Super Saturday");
break;
case 0: document.write("Sleepy Sunday");
break;
default:
document.write("I'm looking forward to this weekend!"); }
</script>
```

The program first looks for a **case** clause with a **label** matching the value of expression. If found, it then transfers control to that clause, executing the associated statements. If no matching label is found, the program looks for the optional default clause. If no default clause is found, the program continues execution at the statement following the end of switch. By convention, the default clause is the last clause.

If break is omitted, the program continues execution at the next statement in the switch statement when there is a matched statement.

Example 2:

```
switch (fruittype) {
  case "Apples":
    document.write("Apples are $0.32 a pound.<br>");
    break;
  case "Bananas":
    document.write("Bananas are $0.48 a pound.<br>");
    break;
  case "Mangoes":
  case "Papayas":
    document.write("Mangoes and papayas are $2.79 a pound.<br>");
    break;
  default:
    document.write("Sorry, we are out of " + fruittype + ".<br>");}
```

Example 3:-

```
<script type="text/javascript">
var grade='A','B';
document.write("Entering switch block<br />");
switch (grade)
{
  case 'A': document.write("Good job<br />");
  case 'B': document.write("Pretty good<br />");
  case 'C': document.write("Passed<br />");
  case 'D': document.write("Not so good<br />");
  case 'F': document.write("Failed<br />");
  default:  document.write("Unknown grade<br />")
}
document.write("Exiting switch block");
</script>
```

B. Working with Loops

A loop is a set of commands that executes repeatedly until a specified condition is met, and JavaScript supports loop like for, do while, and while loop statements. In addition, we can use the break and continue statements within loop statements.

i. for Loop

A for loop repeats until a specified condition evaluates to false. A syntax of for statement is:-

for (*[initialization]*; *[condition]*; *[increment]*){ *Statement(s)*}

Example 1: a program that adds numbers from 0 to 10

```
var counter = 100;
var sum = 0;
 for (var i = 0; i <= counter; i++) {
          sum = sum + i;
 }
document.write("the sum is " + sum);
```

Example 2:-

```
<html> <body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
{
```

```
        if (i==3)
        {
        break;
        }
document.write("The number is " + i); document.write("<br />");
}
 </script>
</body>
</html>
```

ii. do...while Statement

The do...while statement repeats until a specified condition evaluates to false. A do...while statement written as follows:

```
        do{
          statement
        }while (condition);
```

Statement executes once before the condition is **checked**. If condition is true, the statement executes again. At the end of every execution, the condition is checked. When the condition is false, execution stops and control passes to the statement following do...while.

Example: the do loop iterates at least once and reiterates until i is no longer less than 5.

```
        i=1;
        do
        {
          i += 1;
          document.write(i);
        } while (i < 5);
```

iii. while Statement

A while statement executes its statements as long as a specified condition evaluates to true. A while statement looks as follows:

```
        while (condition){
          statement
        }
```

The condition test occurs before statement in the loop are executed. If the condition returns true, statement is executed and the condition is tested again. If the condition returns false, execution stops and control is passed to the statement following while.

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

```
var i=0;
while (i<=10)
{
document.write("The number is " + i);
document.write("<br />");
 i=i+1;
}
</script>
<I>THANK YOU BEING UNDERSTAND ME</I>
</body>
</html>
```

iv. Break, continue and label statements:-

These are used to come out of a loop without reaching at its bottom or to skip a part of the code block and want to start next iteration of the look when needed by users or programmers.

To handle all such situations, JavaScript provides **break, label** and **continue** statements. These statements are used to immediately come out of any loop or to start the next iteration of any loop.

Example1: the following example loops until the value of loop counter is 5:

```
for (i = 0; i < 100; i++) {
if (i== 5)
break;
}
```

The continue statement can be used to restart a while, do-while, for, or label statement. When we use continue, it terminates the current **iteration** and continues execution of the loop with the next iteration. In contrast to the break statement, continue does not terminate the execution of the loop entirely. In a while loop, it jumps back to the condition check, and in a for loop, it jumps to the increment-expression. But, break statement is used to terminate loop.

Example 2: a program that adds numbers between 0 and 100 with the exception of 60, 70, and 80

```
<script type=”text/javascript”>
var counter = 100;
var sum = 0;
 for (var i = 0; i <= counter; i++)
{
        if(i==60 || i==70 || i==80)
                continue;
```

```
            sum = sum + i;
        }
        document.write("the sum is " + sum);
</script>
```

C. Using Arrays

An array is an ordered collection of data. In JavaScript, arrays are limited to a table holding one column of data, with as many rows as needed to hold data.

A JavaScript-enabled browser creates a number of internal arrays for the objects in HTML documents and browser properties. For example, if our document contains five links, the browser maintains a table of those links. To access them by number (with 0 being the first link) in the array **syntax** as in document.links[0], which represents the first link in the document.

- ✓ **To initialize** an array for script, use the **new** keyword to construct the object for assigning the array object to a variable of our choice like:

  var myArray = **new** makeArray(n) where n is the number of entries anticipated for the array. This initialization does not make any array entries or create any **placeholders**.

Example: an array that stores the names of planets

```
planet = new Array(9); //an array with 9 entries
planet[0] = "Mercury"
planet[1] = "Venus"
planet[2] = "Earth"
planet[3] = "Mars"
planet[4] = "Jupiter"
planet[5] = "Saturn"
planet[6] = "Uranus"
planet[7] = "Neptune"
planet[8] = "Pluto"
```

The **index** number starts at **0** and end at **n-1** for array of n entries to access them. For example, to access the fifth planet in the planets array, we use:

planet[4]; //Jupiter

- ✓ **To deleting array entries, we** can always set the value of an array entry to null or an empty string to wipe out any data that used to occupy that space. But with the delete operator, we could not completely remove the element. Deleting an array element eliminates the index from the list of accessible index values but does not reduce the array's length.

```
lanet.length; // result: 9
delete planet[2];
planet.length; //result: 9
```

```
planet[2]; //result: undefined
```

✓ **Array Object Methods**

- **array.concat(array2):-** The array.concat() method allows to join two array objects into a new array object. The action of concatenating the arrays does not alter the contents or behavior of the two original arrays.

For example:

```
var array1 = new Array(1,2,3)
var array2 = new Array("a","b","c")
var array3 = array1.concat(array2)   // result: array with values 1,2,3,"a","b","c"
```

If an array element is a string or number value (not a string or number object), the values are copied from the original arrays into the new one. All connection with the original arrays ceases for those items.

- **array.sort([compareFunction]):-** This methods returns array of entries in the order as determined by the *compareFunction* algorithm.

For simple alphabetical sorting of string values in arrays, the plain array.sort() method does the trick. A comparison function is passed two values from the array. The comparison function lets the sort() method know which of the two items comes before the other, based on the value the function returns. Assuming that the function compares two values, a and b, the returned value reveals information to the sort() method:

Example: a function that sorts numbers

```
<html>
<head>
<title>JavaScript Array sort Method</title>
</head>
<body>
<script type="text/javascript">
var arr = new Array('12', '5', '100', '1', '200', '80');
var sorted = arr.sort();
document.write("Returned string is : " + sorted );
</script>
</body>
</html>
```

Other array object methods

Method	**Description**
every()	Returns true if every element in this array satisfies the provided

	testing function.
filter()	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf(value)	Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
join(separator)	Joins all elements of an array into a string using separator
lastIndexOf(value)	Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.
pop()	Removes the last element from an array and returns that element.
push(value)	Adds one or more elements to the end of an array and returns the new length of the array.
reverse()	Reverses the order of the elements of an array − the first becomes the last, and the last becomes the first.
shift()	Removes the first element from an array and returns that element.
slice(start [,end])	Extracts a section of an array and returns a new array.
splice(start, count)	Adds and/or removes elements from an array.
toString()	Returns a string representing the array and its elements.
unshift(value)	Adds one or more elements to the front of an array and returns the new length of the array.

Table array object methods

Example 1: On some array functions

```
<html>
<head>
  <title>javascript test</title>
<script language="JavaScript">
function arrayFunction() {
        var grade = new Array("70", "60", "80", "90", "20");
        document.write("<br> Popped: " + grade.pop());
        document.write("<br> After poping:");
        for(var i=0;i<grade.length; i++)
                document.write(" " + grade[i]);
```

```
        grade.reverse();
        document.write("<br>Reversed: ");
        for(var i=0;i<grade.length; i++)
                document.write(" " + grade[i]);
        document.write("<br>Search 80: " + grade.indexOf("80"));
        document.write("<br>Converted to string: " + grade.toString());
        document.write("<br>Sliced: " + grade.slice(2));
        document.write("<br>Spliced: " + grade.splice(1));
}
</script>
</head>
<body onLoad="arrayFunction();">
</body>
</html>
```

Example 3:- sort example

```
<html>
<head>
<title>JavaScript Array sort Method</title>
</head>
<body>
<script type="text/javascript">
var arr = new Array("orange", "mango", "banana", "sugar");
var sorted = arr.sort();
document.write("Returned string is : " + sorted );

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

Returned array is : banana,mango,orange,sugar

Example 4:- on push

```
<html>
<head>
<title>JavaScript Array push Method</title>
</head>
<body>
<script type="text/javascript">
var numbers = new Array(1, 4, 9);
var length = numbers.push(10);
document.write("new numbers is : " + numbers );
length = numbers.push(20);
document.write("<br />new numbers is : " + numbers );
</script>
</body>
</html>
```

4.8.JavaScript Functions

Functions are one of the fundamental building blocks in JavaScript. A function is a set of statements that performs a specific task. To use a function, we must first define it; then the script can be called it.

In function definition, it consists of the **function** keyword, followed by:

- The name of the function.
- A list of arguments to the function, enclosed in parentheses and separated by commas.

The JavaScript statements that define the function, enclosed in curly braces, { }. The statements in a function can include calls to other functions defined in the current application. The syntax to define function:

function *functionName* ([*parameter1*]...[,*parameterN*]) {
statement[s]
}

Function names have the same restrictions as variable names. It is possible to use multiword names with the interCap (internally capitalized) format that start with a verb because functions are action items.

Example: the following code defines a simple function named square:

```
function square(number) {
  return number * number;
}
```

The function square takes one argument, called number. The function consists of one statement that indicates to return the argument of the function multiplied by itself. The **return** statement specifies the value returned by the function.

Example:-

```
<html>
<head>
<script type="text/javascript">
// If alert("Hello world!!") below had not been written within a
// function, it would have been executed as soon as the page gets loaded.
function displaymessage() {
alert("Hello World!")
}
</script>
</head>
<body>
<form>
```

```
<input type="button" value="Click me!"
onclick="displaymessage()" >
</form>
</body>
</html>
```

- ✓ **Function Parameters:-**Functions receive parameter values from the calling statement. Parameters (also known as arguments) provide a mechanism for "handing off" a value from one statement to another by way of a function call. If no parameters occur in the function definition, both the function definition and call to the function have only empty sets of parentheses. When a function receives parameters, it assigns the incoming values to the variable names specified in the function definition's parentheses. Consider the following script segment:

```
function sayHiToFirst(first, second, third) {
  alert("Say hello, " + first)
}
sayHiToFirst("Gracie", "George", "Harry")
sayHiToFirst("Larry", "Moe", "Curly")
```

After the function is defined in the script, the next statement calls function, passing three strings as parameters. The function definition automatically assigns the strings to variables first, second, and third. Therefore, before the **alert()** statement inside the function ever runs, first evaluates to "Gracie," second evaluates to "George," and third evaluates to "Harry." In the **alert()** statement, only the first value is used and the alert reads

Say hello, Gracie

When the user closes the first alert, the next call to the function occurs. This time through, different values are passed to the function and assigned to first, second, and third. The alert dialog box reads

Say hello, Larry

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

```
<html>
<head>
<script type="text/javascript">
function product(op1, op2){
    return op1*op2;
}
</script>
```

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

Example:- concatenate , replace, lowercase

```
<html>
<head>
<title>JavaScript String concat() Method</title>
</head>
<body>
<script type="text/javascript">
var str1 = new String( "This is string one" );
var str2 = new String( "This is string two" );
var str3 = str1.concat( str2 );
document.write("Concatenated String :" + str3);
var re = /(\w+)\s(\w+)/;
var str = "zara ali";
var newstr = str.replace(re, "$2, $1");
document.write(newstr);
var str = "Apples are round, and Apples are Juicy.";
document.write(str.toLowerCase( ));
</script>
</body>
<!-- Mirrored from www.tutorialspoint.com/cgi-bin/practice.cgi?file=javascript_76 by HTTrack Website Copier/3.x [XR&CO'2013], Wed, 25 Dec 2013 20:25:32 GMT -->
</html>
```

Managing Events

Events are occurrences generated by the browser, such as loading a document, or by the user, such as moving the mouse. They are the user and browser activities to which we may respond dynamically with a scripting language like JavaScript.

There are several more events that we can capture with JavaScript. Here is a short list of events that Web developers like to capture:

Event	Event Handler	Description
Load	onLoad	Browser finishes loading a Web document
Unload	onUnload	Visitor requests a new document in the browser window
Mouseover	onMouseOver	Visitor moves the mouse over some object in the document window
Mouseout	onMouseOut	Visitor moves the mouse off of some object in the document window
MouseDown	onMouseDown	A mouse button was pressed
MouseMove	onMouseMove	The mouse moved
MouseUp	onMouseUp	The mouse button was released
Select	onSelect	Text has been selected.
Click	onClick	Visitor clicks the mouse button
Focus	onFocus	Visitor gives focus to or makes active a particular window or form element by clicking on it with a mouse or other pointing device or tabbing to it
Blur	onBlur	A form field lost the focus (user moved to another field)
Change	onChange	Visitor changes the data selected or contained in a form element
Submit	onSubmit	Visitor submits a form
Reset	onReset	Visitor resets a form
Abort	onAbort	An image failed to load
Change	onChange	The contents of a field has changed
DblClick	onDblClick	User double-clicked on this item
Error	onError	An error occurred while loading an image
Keydown	onKeyDown	A key was pressed
KeyPress	onKeyPress	A key was pressed OR released
KeyUP	onKeyUp	A key was released

Events are :-

- onabort - Loading of an image is interrupted
  - onblur - An element loses focus
  - onchange - The content of a field changes
  - onclick - Mouse clicks an object
  - ondblclick - Mouse double-clicks an object
  - onerror - An error occurs when loading a document or an image
  - onfocus - An element gets focus
  - onkeydown - A keyboard key is pressed
  - onkeypress - A keyboard key is pressed or held down
  - onkeyup - A keyboard key is released
  - onload - A page or an image is finished loading
  - onmousedown - A mouse button is pressed
  - onmousemove - The mouse is moved
  - onmouseout - The mouse is moved off an element
  - onmouseover - The mouse is moved over an element
  - onmouseup - A mouse button is released
  - onreset - The reset button is clicked
  - onresize - A window or frame is resized
  - onselect - Text is selected
  - onsubmit - The submit button is clicked
  - onunload - The user exits the page

Example 1 :- onblur and onSubmit

```
<html>
<head>
<script type="text/javascript">
function upperCase() {
var x=document.getElementById("fname").value
document.getElementById("fname").value=x.toUpperCase()
}
function validate() {
// return true or false based on validation logic
}
</script>
</head>
<body>
Enter your name:
<input type="text" id="fname" onblur="upperCase()"><br>
<form action="tryjs_submitpage.htm" onsubmit="return validate()">
```

```
Name (max 10 chararcters): <input type="text" id="fname" size="20"><br />
Age (from 1 to 100): <input type="text" id="age" size="20"><br />
E-mail: <input type="text" id="email" size="20"><br />
<br />
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Example 2: a program that adds or subtracts two numbers when respective button is clicked

```
<html>
<head>
<script language="JavaScript">
function adder(num1, num2)
{
        var sum = 0;
        sum = num1 + num2;
        document.write("The sum is " + sum);
}
Function subtractor(num1, num2)
{
        var difference = 0;
        Difference = num1 - num2;
        document.write("The difference is " + difference);
}
</script>
</head>
<body>
<form name="event_example">
<input type="button" value="add" name="add" onClick="adder(10,30)">
<input type="button" value="subtract" name="subtract" onClick="subtractor(20,50)">
</form>
</body>
</html>
```

4.9.Form Processing and Validation

Forms are widely used on the Internet. The form input is often sent back to the server or mailed to a certain e-mail account. With the help of JavaScript the form input can easily be checked before sending it over the Internet. It is sent only if the input is valid.

Form data that typically are checked by a JavaScript could be:

- Has the user left required fields empty?
- Has the user entered a valid e-mail address?

- Has the user entered a valid date?
- Has the user entered text in a numeric field?

In the name field, we will receive an error message when nothing is entered. Age is number and we expect a positive number only. If user enters characters which are not number, or negative value, it is not valid age. So the script should check this and determine its validity.

The email field is a little bit more sophisticated. It shouldn't work if there is no @ symbol in the email because valid email addresses contain that symbol. The criteria for accepting the input as a valid e-mail address is the @. The person may enter wrong emails which could pass as valid but this is the least check we could do.

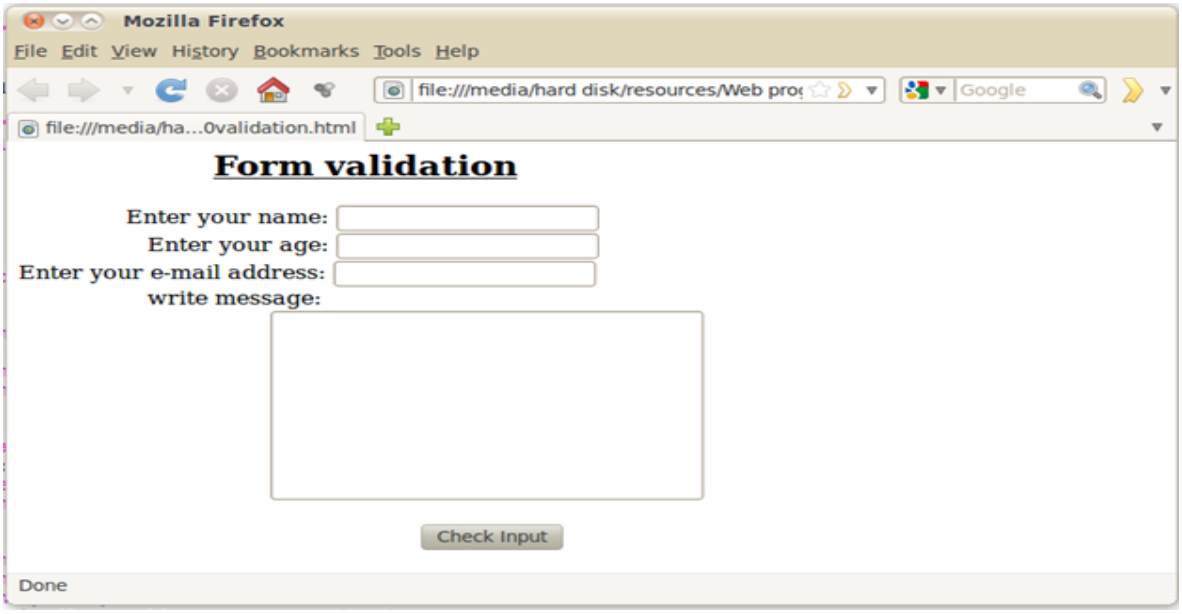


Figure form validation

Example: a script that creates and validates the above form:

```
<html>
<head>
<script language="JavaScript">
function check(form)
{
if (form.urname.value == "")
   alert("Please enter a string as your name!")
else
   alert("Hi " + form.urname.value + "! Name ok!");
if(form.age.value < 0 || form.age.value=="")
  alert("Age should be number and greater than 0");
else
  alert("Age ok");
if (form.email.value == "" || form.email.value.indexOf('@', 0) == -1)
```

```
    alert("No valid e-mail address!");
else
    alert("Email oK!");
if(form.urmessage.value==="")
   alert("No message written");
else
   alert("Message ok");
}
</script>
</head>
<body>
<h2>  <u> Form validation </u> </h2>
<form name="first">
Enter your name: <input type="text" name="urname"> <br>
Enter your age: <input type="text" name="age"> <br>
Enter your e-mail address: <input type="text" name="email"> <br>
write message: <textarea name="urmessage" cols=40 rows=10></textarea><br><br>
<input type="button" name="validate" value="Check Input"
onClick="check(this.form)">
</body>
</html>
```

- **E-mail Validation** The function below checks if the content has the general syntax of an email. For example, the input data must contain at least an @ sign and a dot (.). Also, the @ must not be the first character of the email address, and the last dot must at least be one character after the @ sign. The entire script, with the HTML form could look like:

```
<html>
 <head>
<script type="text/javascript">
function validate_email(field,alerttxt)
{
 with (field)
{
apos=value.indexOf("@");
dotpos=value.lastIndexOf(".");
if (apos<1||dotpos-apos<2)
{alert(alerttxt);return false;}
 else
{  return true;
} } }
function validate_form(thisform)
 {
 with (thisform)
{
 If
 (validate_email(email,"Not a valid e-mail address!")==false)
{email.focus();return false;}
} }
</script>
</head>
<body>
<form action="submitpage.htm"
 onsubmit="return validate_form(this);"
 method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
 </body>
```

```
</html>
```

Data Format Validation:

This example shows how to validate an entered email address which means email address must contain at least an @ sign and a dot (.). Also, the @ must not be the first character of the email address, and the last dot must at least be one character after the @ sign:

```
<script type="text/javascript">
<!--
function validateEmail()
{
  var emailID = document.myForm.EMail.value;
  atpos = emailID.indexOf("@");
  dotpos = emailID.lastIndexOf(".");
  if (atpos < 1 || ( dotpos - atpos < 2 ))
  {
     alert("Please enter correct email ID")
     document.myForm.EMail.focus ();
     return false;
  }
  return( true );
}
//--></script>
```

4.10. JavaScript Animation

It is possible to use JavaScript to create animated images. The trick is to let a JavaScript change between different images on different events. In the following example we will add an image that should act as a link button on a web page. We will then add an onMouseOver event and an onMouseOut event that will run two JavaScript functions that will change between the images.

Examples: - Button animation

```
<html> <head>
<script type="text/javascript">
function mouseOver() { document.b1.src ="b_blue.gif"; }
 function mouseOut() { document.b1.src ="b_pink.gif"; } </script> </head>
<body>
<a href="http://www.w3schools.com" target="_blank">
<img border="0"  src="b_pink.gif" name="b1" width="26" height="26"
onmouseover="mouseOver()" onmouseout="mouseOut()" />
```

</a> </body> </html>

The HTML Code

The HTML code looks like this:

```
<a href="http://www.w3schools.com" target="_blank">
<img border="0" src="b_pink.gif" name="b1"
onmouseOver="mouseOver()"
onmouseOut="mouseOut()" />  </a>
```

Note that we have given the image a name to make it possible for JavaScript to address it later. The onMouseOver event tells the browser that once a mouse is rolled over the image, the browser should execute a function that will replace the image with another image.

The onMouseOut event tells the browser that once a mouse is rolled away from the image, another JavaScript function should be executed. This function will insert the original image again.

We can use JavaScript to create a complex animation which includes but not limited to:

- Fireworks
- Fade Effect
- Roll-in or Roll-out
- Page-in or Page-out
- Object movements

We might be interested in existing JavaScript based animation library. JavaScript can be used to move a number of DOM elements (<img />, or any other HTML element) around the page according to some sort of pattern determined by a logical equation or function. JavaScript provides following two functions to be frequently used in animation programs.

- **setTimeout (function, duration)** - This function calls *function* after *duration* milliseconds from now.
- **setInterval(function, duration)** - This function calls *function* after every *duration* milliseconds.
- **clearTimeout(setTimeout_variable)** - This function calls clears any timer set by the setTimeout() functions.

JavaScript can also set a number of attributes of a DOM object including its position on the screen. You can set *top* and *left* attribute of an object to position it anywhere on the screen. Here is the simple syntax:

```
// Set distance from left edge of the screen.
```

object.style.left = distance in pixels or points;

or // Set distance from top edge of the screen.

object.style.top = distance in pixels or points;

Manual Animation:

So let's implement one simple animation using DOM object properties and JavaScript functions as follows:

```
<html>
<head>
<title>JavaScript Animation</title>
<script type="text/javascript">
<!--
var imgObj = null;
function init(){
  imgObj =
document.getElementById('myImage');
  imgObj.style.position= 'relative';
  imgObj.style.left = '0px'; }
function moveRight(){
  imgObj.style.left =
parseInt(imgObj.style.left) + 10 + 'px';
}
window.onload =init;
//-->
</script>
</head>
<body>
<form>
<img id="myImage" src="/images/html.gif" />
<p>Click button below to move the image to right</p>
<input type="button" value="Click Me" onclick="moveRight();" />
</form>
</body>
</html>
```

Here is the explanation of the above example:

- We are using JavaScript function *getElementById()* to get a DOM object and then assigning it to a global variable *imgObj*.
- We have defined an initialization function *init()* to initialize *imgObj* where we have set its *position* and *left* attributes.
- We are calling initialization function at the time of window load.
- Finally, we are calling *moveRight()* function to increase left distance by 10 pixels. You could also set it to a negative value to move it to the left side

Automated Animation:

In the above example we have seen , how an image moves to right with every click. We can automate this process by using JavaScript function ***setTimeout()*** as follows:

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

```
<title>JavaScript Animation</title>
<script type="text/javascript">
<!--
var imgObj = null;
var animate ;
function init(){
  imgObj =
document.getElementById('myImage');
  imgObj.style.position= 'relative';
  imgObj.style.left = '0px';
}
function moveRight(){
  imgObj.style.left =
parseInt(imgObj.style.left) + 10 + 'px';
  animate = setTimeout(moveRight,20); // call
moveRight in 20msec
}
function stop(){
  clearTimeout(animate);
  imgObj.style.left = '0px';
}
window.onload =init;
//-->
</script>
</head>
<body>
<form>
<img id="myImage" src="/images/html.gif" />
<p>Click the buttons below to handle
animation</p>
<input type="button" value="Start"
onclick="moveRight();" />
<input type="button" value="Stop"
onclick="stop();" />
</form>
</body>
</html>
```

Here we have add more spice. So let's see what is new here:

- The ***moveRight()*** function is calling ***setTimeout()*** function to set the position of ***imgObj.***
- We have added a new function ***stop()*** to clear the timer set by ***setTimeout()*** function and to set the object at its initial position.

Here are 20 multiple-choice questions about HTML along with their answers:

1. What does HTML stand for?

a) Hyper Text Markup Language [Answer]

b) Home Tool Markup Language

c) Hyperlinks and Text Markup Language

d) Hyper Transfer Markup Language

2. Which tag is used to define a hyperlink?

a) <p>

b) <a> [Answer]

c) <h1>

d) <b>

3. What is the correct HTML tag for the largest heading?

a) <h5>

b) <h1> [Answer]

c) <head>

d) <header>

4. Which tag is used to create a line break in HTML?

a) <lb>

**b)
 [Answer]**

c) <break>

d) <bl>

5. What is the correct HTML for inserting an image?

a) <img src="image.jpg">

b) <image src="image.jpg">

c) <img href="image.jpg">

d) <img src="image.jpg"> [Answer]

6. Which tag is used to define an unordered list in HTML?

a) <ol>

b) <ul> [Answer]

c) <li>

d) <u>

7. Which tag is used to define a table row in HTML?

a) <td>

b) <tr> [Answer]

c) <th>

d) <table>

8. Which tag is used to define a footer in HTML?

a) <foot>

b) <footer> [Answer]

c) <end>

d) <endfooter>

9. Which tag is used to highlight a text in HTML?

a) <highlight>

**b) **

c) <mark> [Answer]

**d) **

10. What is the correct HTML for creating a hyperlink that opens in a new tab?

a) <a href="url" target="_new">

b) <a href="url" target="_blank"> [Answer]

c) <a href="url" target="_tab">

d) <a href="url" target="_window">

11. Which tag is used to define an input field in HTML?

a) <text>

b) <input> [Answer]

c) <field>

d) <textbox>

12. What does the HTML <canvas> element allow you to do?

a) Embed external media files

b) Display scalable vector graphics

c) Draw graphics and animations [Answer]

d) Modify font styles

13. In HTML, which attribute is used to specify a unique name for an element?

a) id [Answer]

b) class

c) name

d) style

14. Which HTML tag is used to define a navigation menu?

a) <nav> [Answer]

b) <menu>

c) <navigation>

d) <navbar>

15. What is the correct HTML for creating a checkbox?

a) <check>

b) <checkbox>

c) <input type="check"> [Answer]

d) <input type="checkbox">

16. Which attribute is used to specify the URL of the linked resource in the <a> tag?

a) src

b) url

c) link

d) href [Answer]

17. Which tag is used to add a horizontal line in HTML?

a) <hr> [Answer]

b) <line>

c) <break>

d) <horizontal>

18. Which tag is used to define the header of a table or a section?

a) <header>

b) <heading>

c) <th>

d) <h1> [Answer]

19. Which attribute is used to specify an alternative text for an image in HTML?

a) alt [Answer]

b) src

c) title

d) caption

20. What is the correct HTML for creating an email link?

a) <a href="mailto:info@example.com"> [Answer]

b) <a href="email:info@example.com">

c) <a href="https://example.com">

d) <a href="emailto:info@example.com">

Here are 20 HTML questions along with their answers:

1. What does HTML stand for?

HTML stands for HyperText Markup Language.

2. What is the purpose of HTML?

HTML is used to structure and present the content on the World Wide Web.

3. What are the basic building blocks of HTML?

HTML consists of elements, which are represented by tags.

4. How do you create a hyperlink in HTML?

You can create a hyperlink in HTML using the anchor tag (<a>) and the href attribute.

5. What is the default behavior of an HTML button?

By default, an HTML button triggers a form submission or a page reload.

6. How do you add comments in HTML?

HTML comments can be added using the <!-- and --> syntax.

7. What is the difference between inline and block-level elements?

Inline elements do not start on a new line and only occupy the space necessary for their content. Block-level elements, on the other hand, start on a new line and occupy the full width available.

8. What is the purpose of the <head> tag in HTML?

The <head> tag is used to define the head section of an HTML document, which contains information about the document such as the title, stylesheets, scripts, etc.

9. How do you create a numbered list in HTML?

You can create a numbered list using the <ol> (ordered list) tag.

10. How do you add an image in HTML?

You can add an image using the <img> tag and specifying the source (src) attribute.

11. What is the purpose of the alt attribute in an <img> tag?

The alt attribute is used to provide alternative text for an image, which is displayed if the image fails to load.

12. What is the difference between padding and margin in CSS?

Padding is the space inside an element, whereas margin is the space outside the element.

13. How do you style an HTML element using CSS?

You can style an HTML element using CSS by applying styles to the element's selector in a CSS rule.

14. What is the purpose of the tag in HTML?

The tag is used as a container or a block-level element to group and style content.

15. How do you create a table in HTML?

You can create a table using the <table> tag along with additional tags like <tr> (table row) and <td> (table data).

16. What is the purpose of the alt attribute in an <a> tag?

The alt attribute in an <a> tag is used to provide alternative text for the link, which is displayed when the link is hovered over or if the link fails to open.

17. What are the different types of lists in HTML?

HTML provides three types of lists: ordered lists (<ol>), unordered lists (<ul>), and definition lists (<dl>).

18. What is the difference between a class and an ID in CSS?

While both class and ID are used for styling elements, a class can be used by multiple elements, whereas an ID is unique to a single element.

19. What is the purpose of the tag in HTML?

The tag is an inline-level element used to style a certain portion of text within a block-level element.

20. How do you embed external content, such as a video, in HTML?

You can embed external content, like a video, using the <iframe> tag and providing the source (src) attribute.