THE NEW COLLEGE (AUTONOMOUS)

Sponsored by: The Muslim Educational Association of Southern India

(Affiliated to the Madras University & Re-Accredited by NAAC with 'A' Grade)

Chennai-600014

INTERNET AND WEB DESIGN

Subject Code: 17BRM307

Semester-V

Date: 15-07-2018

Batch: 2018-2019

Prepared By

Prof. N. ANVER HUSSAIN

Department of Information Systems Management, Shift-II

The New College (Autonomous)

Chennai-600014

CORE – 7: INTERNET AND WEB DESIGN

E-Notes

Subject Code : 17BRM 307

Duration : 3 Hours Max Marks: 75

Unit 1:

1.Introduction to Internet: - Definition

The **Internet** (or **internet**) is the global system of interconnected computer networks that uses the **Internet** protocol suite (TCP/IP) to communicate between networks and devices.

2. History of Internet

The Internet has revolutionized the computer and communications world like nothing before. The invention of the telegraph, telephone, radio, and computer set the stage for this unprecedented integration of capabilities. The Internet is at once a world-wide broadcasting capability, a mechanism for information dissemination, and a medium for collaboration and interaction between individuals and their computers without regard for geographic location. The Internet represents one of the most successful examples of the benefits of sustained investment and commitment to research and development of information infrastructure. Beginning with the early research in packet switching, the government, industry and academia have been partners in evolving and deploying this exciting new technology. Today, terms like "bleiner@computer.org" and "http://www.acm.org" trip lightly off the tongue of the random person on the street. 1

This is intended to be a brief, necessarily cursory and incomplete history. Much material currently exists about the Internet, covering history, technology, and usage. A trip to almost any bookstore will find shelves of material written about the Internet.

In this paper,3 several of us involved in the development and evolution of the Internet share our views of its origins and history. This history revolves around four distinct aspects. There is the technological evolution that began with early research on packet switching and the ARPANET (and related technologies), and where current research continues to expand the horizons of the infrastructure along several dimensions, such as scale, performance, and higher-level functionality. There is the operations and management aspect of a global and complex operational infrastructure. There is the social aspect, which resulted in a broad community of Internets working together to create and evolve the technology. And there is the commercialization aspect, resulting in an extremely effective transition of research results into a broadly deployed and available information infrastructure.

The Internet today is a widespread information infrastructure, the initial prototype of what is often called the National (or Global or Galactic) Information Infrastructure. Its history is complex and involves many aspects – technological, organizational, and community. And its influence reaches not only to the technical fields of computer communications but throughout society as we move toward increasing use of online tools to accomplish electronic commerce, information acquisition, and community operations.

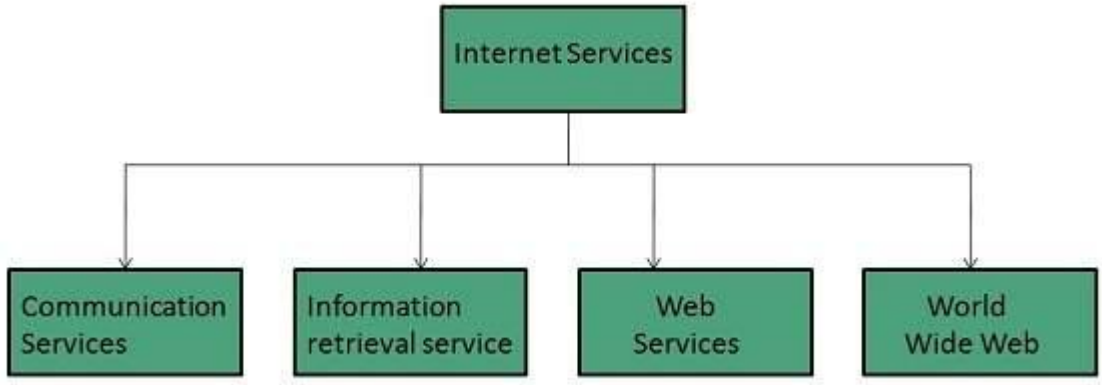
3. Internet Services and Accessibility

Internet Services

Previous Page

Next Page

Internet Services allows us to access huge amount of information such as text, graphics, sound and software over the internet. Following diagram shows the four different categories of Internet Services.



Communication Services

There are various Communication Services available that offer exchange of information with individuals or groups. The following table gives a brief introduction to these services:

S.N.	Service Description
1	**Electronic Mail** Used to send electronic message over the internet.
2	**Telnet** Used to log on to a remote computer that is attached to internet.

3	**Newsgroup** Offers a forum for people to discuss topics of common interests.
4	**Internet Relay Chat (IRC)** Allows the people from all over the world to communicate in real time.
5	**Mailing Lists** Used to organize group of internet users to share common information through e-mail.
6	**Internet Telephony (VoIP)** Allows the internet users to talk across internet to any PC equipped to receive the call.
7	**Instant Messaging** Offers real time chat between individuals and group of people. Eg. Yahoo messenger, MSN messenger.

Information Retrieval Services

There exist several Information retrieval services offering easy access to information present on the internet. The following table gives a brief introduction to these services:

S.N.	Service Description
1	**File Transfer Protocol (FTP)** Enable the users to transfer files.
2	**Archie** It's updated database of public FTP sites and their content. It helps to search a file by its name.
3	**Gopher** Used to search, retrieve, and display documents on remote sites.
4	**Very Easy Rodent Oriented Netwide Index to Computer Achieved (VERONICA)** VERONICA is gopher based resource. It allows access to the information resource stored

	on gopher’s servers.

Web Services

Web services allow exchange of information between applications on the web. Using web services, applications can easily interact with each other.

The web services are offered using concept of **Utility Computing.**

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.

Video Conferencing

Video conferencing or Video teleconferencing is a method of communicating by two-way video and audio transmission with help of telecommunication technologies.

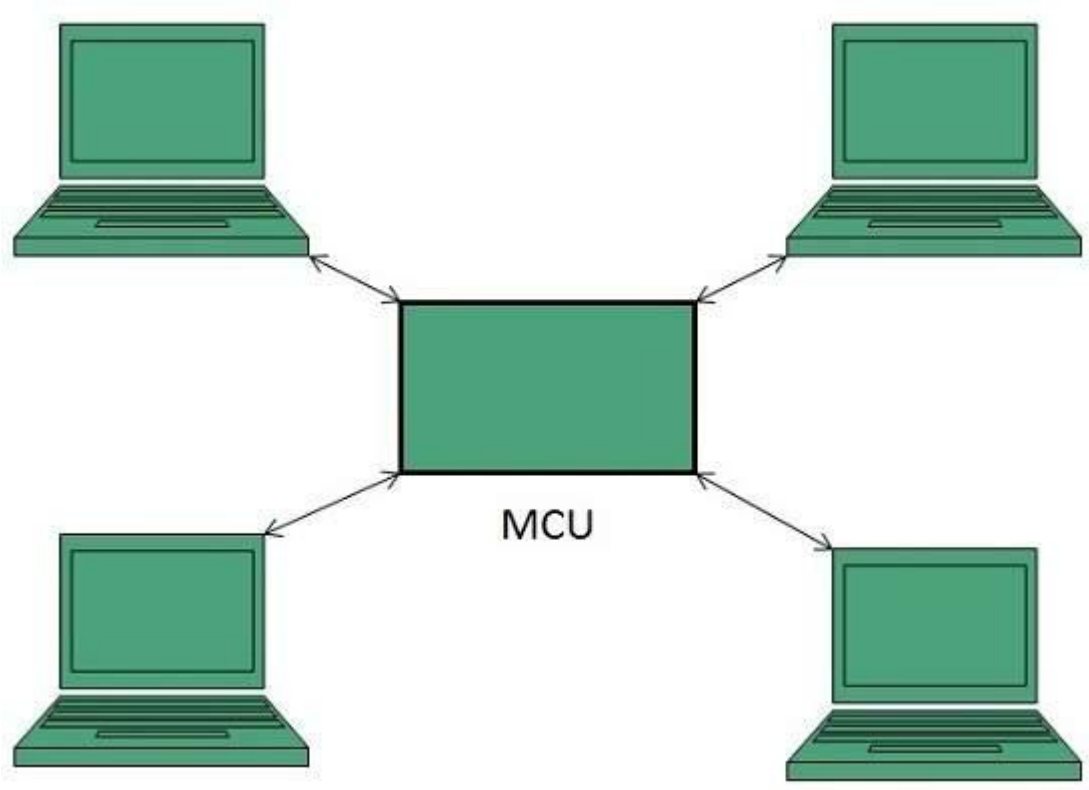
Modes of Video Conferencing

Point-to-Point

This mode of conferencing connects two locations only.

Multi-point

This mode of conferencing connects more than two locations through **Multi-point Control Unit (MCU).**



4. Uses of Internet-Protocols

IP is the core of the TCP/IP protocol suite. IP provides the fundamental mechanism using which data is delivered between devices which may or may not be in the same network.

Addressing

While sending datagrams, an addressing mechanism is needed to send the datagrams accurately. In order to achieve this, IP uses a technique for host addressing. The addressing of devices (to which the datagrams are delivered) needs to be unique as this system needs to work across networks.

Routing

When a datagram is sent from one network to another, which are distant and not directly connected, the delivery is indirect. IP supports this functionality by routing the datagram through intermediate devices (routers). It uses Internet Control Message Protocol (ICMP) and routing protocols like Routing Information Protocol (RIP) and Border Gateway Protocol (BGP) to achieve this.

DATA Encapsulation

IP provides security to networks by encapsulating the data into an IP datagram. This makes sure it is received and interpreted by the intended recipient.

Formatting/Packaging

IP uses a specific formatting and packaging prior to transmission. IP accepts data from the transport layer protocols above it in the OSI layer--UDP and TCP--and passes them to the data layers. This format and package is only decipherable by the recipient.

Fragmentation

Since the frame size of each physical and data link network using IP may be different, IP fragments datagrams into pieces, so that they can each be carried on the local network. This helps with network reliability.

Reassembly

IP reassembles the datagrams received into the full IP datagram (as they may be fragmented) for the receiving device and passes it on to the higher layers for interpretation.

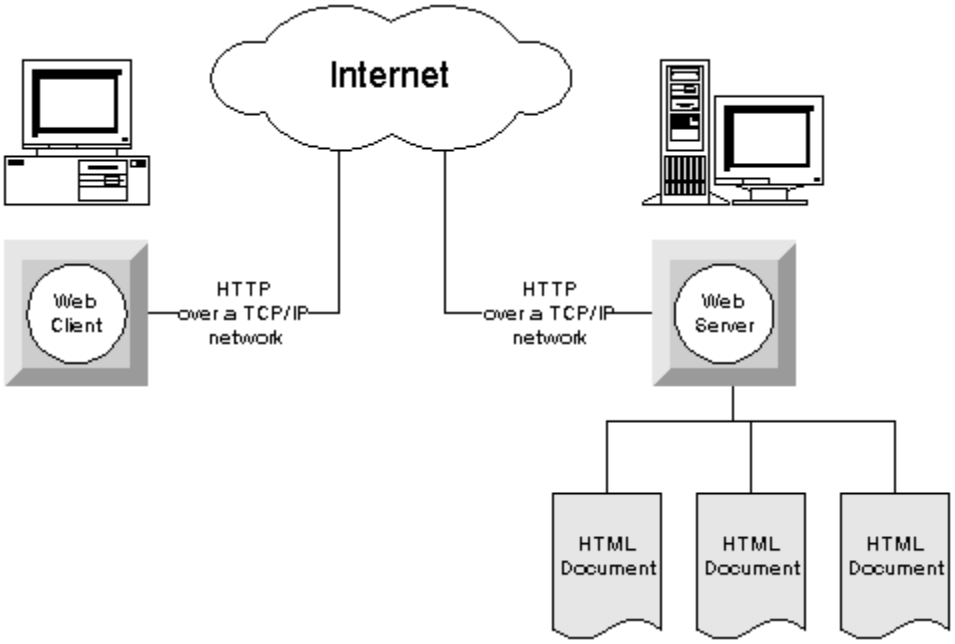
5. Web Concepts:

Basic Web concepts

Who needs to read this section?

If you are already familiar with the terms TCP/IP, HTTP, Web servers, Web clients, HTML, URL, Intranet, and Internet, and you know how information is stored and transmitted over the Web, you don't need to read this section.

Storing, locating, and transmitting information on the Web



Clients, servers, and sites

On the Web, information is stored at **Web sites**. Access to the information at a Web site is managed by a **Web server** for the site. Users access the information using a **Web client**, which is also called a **browser**.

How information is stored: HTML

Information on the Web is stored in documents, using a language named **HTML** (HyperText Markup Language). Web clients interpret HTML and display the documents to a user. The protocol that governs the exchange of

information between the Web server and Web client is named **HTTP** (HyperText Transfer Protocol).

How information is located: the URL

To move from one page of a document to another page, or to another document on the same or another Web site, the user clicks a link in the document shown in their Web client. Documents and locations within documents are identified by an address that is called a Uniform Resource Locator, or **URL**. This is an example of a URL:

```
http://www.sybase.com/products
```

URLs contain information about which server the document is on, and may also specify a particular document available to that server, and even a position within the document. In addition, a URL may carry other information from a Web client to a Web server, including the values entered into fields in an HTML form.

For more information about URLs and addresses on the Web, see the material on the World Wide Web Consortium pages, at the following address:

```
http://www.w3.org/pub/WWW/Addressing/
```

When a user clicks a link in a document on their Web client, the URL is sent to the server of the indicated Web site. The Web server locates the document, and sends the HTML across the network to the Web client.

External data in HTML documents

HTML documents can reference external files (for example, a GIF or JPEG file for a graphic). Not all these external formats are supported by all Web clients. When the document contains such data, the Web client can send a request to the Web server to provide the relevant graphic. If the Web client does not support the format, it does not request the information from the server.

Problems with file-based Web sites

In a file-based Web site, each document is a separate file. For large Web sites, this leads to management problems. For example, maintaining current copies of hundreds or thousands of different resources in separate files is difficult enough; maintaining the links between these resources is even more challenging.

Another problem is that many Web sites contain dynamic information - pricing information, for example, or employee information on an organization's intranet.

Maintaining such information in HTML files in addition to the database where it resides is a huge task.

For these and other reasons, linking databases to the Web is increasingly the solution of choice for management of large Web sites and management of dynamic content. Database storage of Web information can either replace or complement file storage of Web resources.

6. Internet Protocols:

Transmission Control Protocol (TCP)

TCP is a connection oriented protocol and offers end-to-end packet delivery. It acts as back bone for connection.It exhibits the following key features:

- Transmission Control Protocol (TCP) corresponds to the Transport Layer of OSI Model.
- TCP is a reliable and connection oriented protocol.
- TCP offers:
    - Stream Data Transfer.
    - Reliability.
    - Efficient Flow Control
    - Full-duplex operation.
    - Multiplexing.
- TCP offers connection oriented end-to-end packet delivery.
- TCP ensures reliability by sequencing bytes with a forwarding acknowledgement number that indicates to the destination the next byte the source expect to receive.
- It retransmits the bytes not acknowledged with in specified time period.

TCP Services

TCP offers following services to the processes at the application layer:

- Stream Delivery Service
- Sending and Receiving Buffers
- Bytes and Segments
- Full Duplex Service
- Connection Oriented Service
- Reliable Service

Stream Deliver Service

TCP protocol is stream oriented because it allows the sending process to send data as stream of bytes and the receiving process to obtain data as stream of bytes.

Sending and Receiving Buffers

It may not be possible for sending and receiving process to produce and obtain data at same speed, therefore, TCP needs buffers for storage at sending and receiving ends.

Bytes and Segments

The Transmission Control Protocol (TCP), at transport layer groups the bytes into a packet. This packet is called segment. Before transmission of these packets, these segments are encapsulated into an IP datagram.

Full Duplex Service

Transmitting the data in duplex mode means flow of data in both the directions at the same time.

Connection Oriented Service

TCP offers connection oriented service in the following manner:

1. TCP of process-1 informs TCP of process – 2 and gets its approval.
2. TCP of process – 1 and TCP of process – 2 and exchange data in both the two directions.
3. After completing the data exchange, when buffers on both sides are empty, the two TCP's destroy their buffers.

Reliable Service

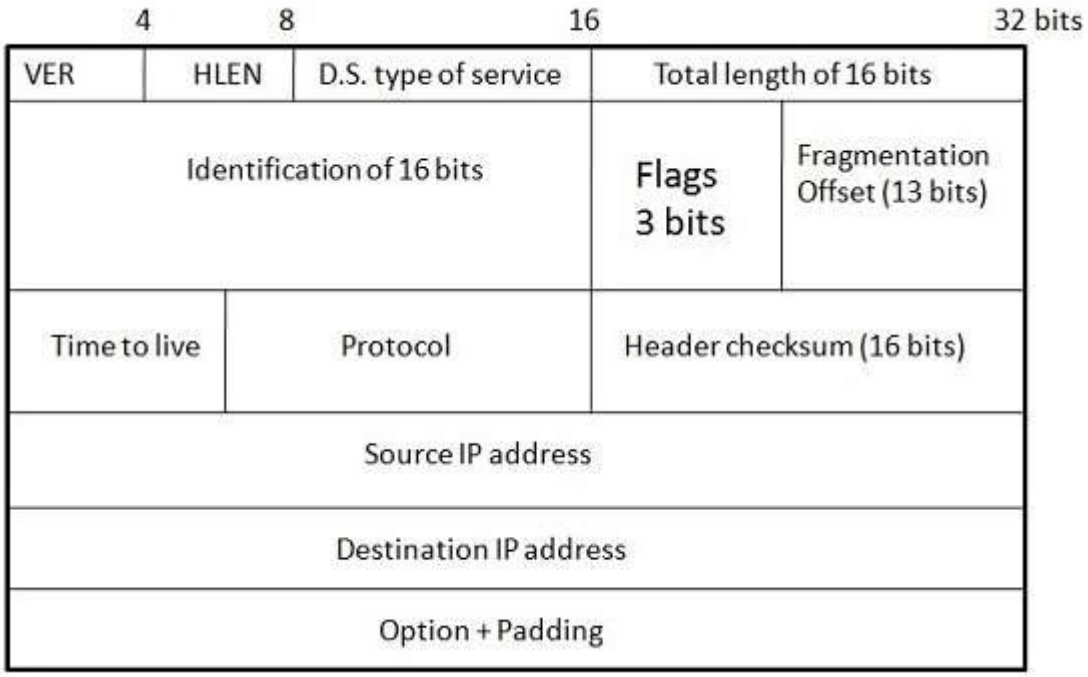
For sake of reliability, TCP uses acknowledgement mechanism.

Internet Protocol (IP)

Internet Protocol is **connectionless** and **unreliable** protocol. It ensures no guarantee of successfully transmission of data.

In order to make it reliable, it must be paired with reliable protocol such as TCP at the transport layer.

Internet protocol transmits the data in form of a datagram as shown in the following diagram:



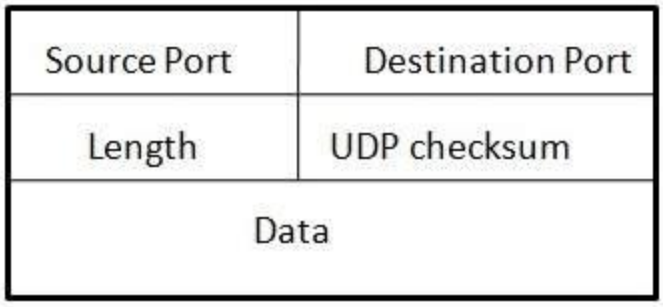
Points to remember:

- The length of datagram is variable.
- The Datagram is divided into two parts: **header** and **data.**
- The length of header is 20 to 60 bytes.
- The header contains information for routing and delivery of the packet.

User Datagram Protocol (UDP)

Like IP, UDP is connectionless and unreliable protocol. It doesn't require making a connection with the host to exchange data. Since UDP is unreliable protocol, there is no mechanism for ensuring that data sent is received.

UDP transmits the data in form of a datagram. The UDP datagram consists of five parts as shown in the following diagram:



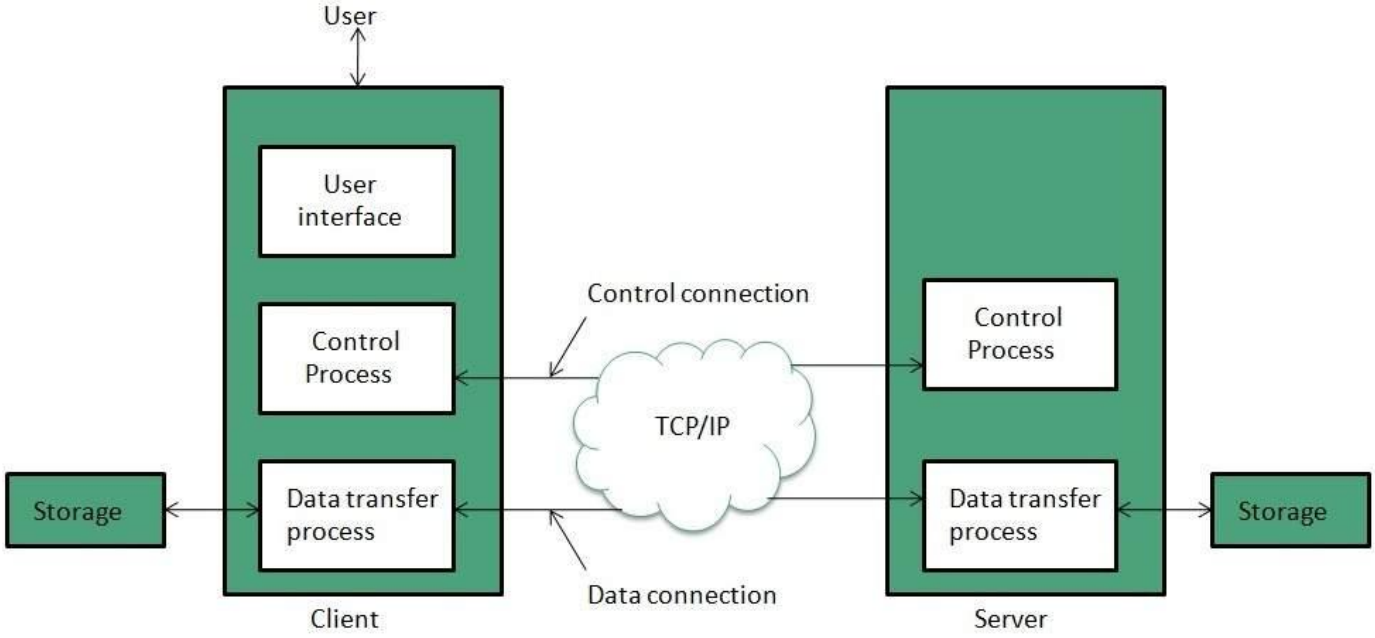
Points to remember:

- UDP is used by the application that typically transmit small amount of data at one time.
- UDP provides protocol port used i.e. UDP message contains both source and destination port number, that makes it possible for UDP software at the destination to deliver the message to correct application program.

File Transfer Protocol (FTP)

FTP is used to copy files from one host to another. FTP offers the mechanism for the same in following manner:

- FTP creates two processes such as Control Process and Data Transfer Process at both ends i.e. at client as well as at server.
- FTP establishes two different connections: one is for data transfer and other is for control information.
- **Control connection** is made between **control processes** while **Data Connection** is made between<="" b="">
- FTP uses **port 21** for the control connection and **Port 20** for the data connection.



Trivial File Transfer Protocol (TFTP)

Trivial File Transfer Protocol is also used to transfer the files but it transfers the files without authentication. Unlike FTP, TFTP does not separate control and data information. Since there is no authentication exists, TFTP lacks in security features therefore it is not recommended to use TFTP.

Key points

- TFTP makes use of UDP for data transport. Each TFTP message is carried in separate UDP datagram.

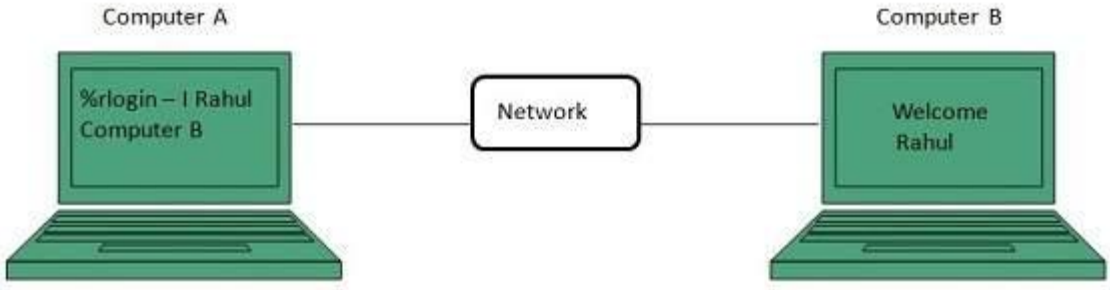
- The first two bytes of a TFTP message specify the type of message.
- The TFTP session is initiated when a TFTP client sends a request to upload or download a file.
- The request is sent from an ephemeral UDP port to the **UDP port 69** of an TFTP server.

Difference between FTP and TFTP

S.N.	Parameter	FTP	TFTP
1	Operation	Transferring Files	Transferring Files
2	Authentication	Yes	No
3	Protocol	TCP	UDP
4	Ports	21 – Control, 20 – Data	Port 3214, 69, 4012
5	Control and Data	Separated	Separated
6	Data Transfer	Reliable	Unreliable

Telnet

Telnet is a protocol used to log in to remote computer on the internet. There are a number of Telnet clients having user friendly user interface. The following diagram shows a person is logged in to computer A, and from there, he remote logged into computer B.



Hyper Text Transfer Protocol (HTTP)

HTTP is a communication protocol. It defines mechanism for communication between browser and the web server. It is also called request and response protocol because the communication between browser and server takes place in request and response pairs.

HTTP Request

HTTP request comprises of lines which contains:

- Request line
- Header Fields
- Message body

Key Points

- The first line i.e. the **Request line** specifies the request method i.e. **Get** or **Post.**
- The second line specifies the header which indicates the domain name of the server from where index.htm is retrieved.

HTTP Response

Like HTTP request, HTTP response also has certain structure. HTTP response contains:

- Status line
- Headers
- Message body

7. IP address

An **Internet Protocol address** (**IP address**) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication.[1][2] An IP address serves two main functions: host or network interface identification and location addressing.

Internet Protocol version 4 (IPv4) defines an IP address as a 32-bit number.[2] However, because of the growth of the Internet and the depletion of available IPv4 addresses, a new version of IP (IPv6), using 128 bits for the IP address, was standardized in 1998.[3][4][5] IPv6 deployment has been ongoing since the mid-2000s.

IP addresses are written and displayed in human-readable notations, such as *172.16.254.1* in IPv4, and *2001:db8:0:1234:0:567:8:1* in IPv6. The size of the routing prefix of the address is designated in CIDR notation by suffixing the address with the number of significant bits, e.g., *192.168.1.15/24*, which is equivalent to the historically used subnet mask *255.255.255.0*.

The IP address space is managed globally by the Internet Assigned Numbers Authority (IANA), and by five regional Internet registries (RIRs) responsible in their designated territories for assignment to local Internet registries, such as Internet service providers (ISPs), and other end users. IPv4 addresses were distributed by IANA to the RIRs in blocks of approximately 16.8 million addresses each, but have been exhausted at the IANA level since 2011. Only one of the RIRs still has a supply for local assignments in Africa.[6] Some IPv4 addresses are reserved for private networks and are not globally unique.

Network administrators assign an IP address to each device connected to a network. Such assignments may be on a *static* (fixed or permanent) or *dynamic* basis, depending on network practices and software features.

8. Types of Internet Address

The Different Categories of IP Addresses

There aren't just two types of IP addresses. Actually, there are a number of different categories that each include a couple of different types of IP addresses. Sounds complicated, right?

It's not that hard to understand when it's laid out in clear terms. Here's what you need to know.

The 2 Types of IP Addresses Consumers Have

Every person or business with an internet service plan will have two types of IP addresses: their private IP addresses, and the public IP address.

Private IP Addresses

Every device that connects to your home internet network has a private IP address. Most obviously, that will include any computers, smartphones, and tablets used in your household. But it also likely includes any bluetooth devices you use like speakers or printers, any smart products you've set up, any smart TVs or Rokus you have, and so on. With a growing industry of internet of things (IoT) products, the number of private IP addresses you're likely to have in your own home is growing.

Your router needs a way to identify each of these items separately, and many of the items need a way to recognize each other. For example, when you go to pair your bluetooth headphones with your smartphone, you need the products to recognize each other. Your router therefore generates private IP addresses which are unique identifiers for each product that differentiate them all on the network.

There are ways to find out what the private IP is for most of the devices on your network if you want to know, and ways to change your IP address on most devices as well. But for the most part, you won't need to know what private IP address each item has. That's information your router will use, but most of the routing devices connected to your network interface will have more recognizable names for anytime you need to find them, ones that either you or the manufacturer supplied.

Public IP Addresses

Your public IP address is the one primary address associated with your whole network. Where each of the connected devices has their own IP address, all of them are also included under the main public IP address for your network.

Your public IP address is provided to your router by your internet service provider (ISP). They generally have a large pool of IP addresses they've bought and distributed among their different

customers. This is the address that all the devices outside of your internet network will use to recognize your network and anything connected to it. This is also the address you saw if you clicked on the page we provided from HostGator to learn your IP address up above.

Your public IP address is tied to all your internet activity. If ISPs get a notice about illegal activity happening on their servers, the public IP address tells them which customer is behind it. So if you download media illegally or send emails that count as spam, your public IP address is how those behaviors can be tracked back to you.

The 2 Types of Public IP Addresses

Public IP addresses come in two main varieties of their own.

Dynamic IP Addresses

Dynamic IP addresses automatically change frequently. Most internet users will be provided with a dynamic IP address from their ISP. ISPs buy a big block of IP addresses and automatically assign one to each customer. Then, they'll periodically assign them a new one and put the older one back in the pool to be recycled for another customer.

That may seem like a strange approach to providing IP addresses, but it works out to be cheaper for ISPs and makes general maintenance easier. When the movement of IP addresses between different networks is a common and automated process, nothing special has to be done to re-set up an IP address for a customer after they make a move.

It also works out to being more secure for customers. When your IP address changes regularly, it's that much harder for outsiders to hack into your network interface. For most internet users, a dynamic address makes the most sense, but there are exceptions.

Static IP Addresses

Static IP addresses are consistent. A network is assigned one once, and you can count on it to stay the same over months and years. Not many people or businesses need static IP addresses, but there are some rare use cases where going with a static IP address is important. The main one is for any business that plans to host their own server.

If you'll be maintaining your own server, then a static IP address will ensure that any websites or email addresses on it will be tied to consistent IP addresses. That's important if you want other devices to be able to consistently find them on the web.

For most people and businesses though, having your own server doesn't make sense. It's much more affordable and convenient to go with a web hosting company that already owns plenty of servers and has the resources needed to maintain them safely and effectively.

The 2 Types of Website IP Addresses

If you're planning to start a website and you do go with a web hosting service provider (which is recommended for almost all website owners), you'll potentially encounter two types of website IP address options.

Shared IP Addresses

The most common types of hosting plans that website owners opt for involve sharing a server with other websites. With shared hosting plans—the most common and affordable option—your website will typically be one of dozens, if not hundreds of websites hosted on the same server.

That works out fine for a lot of websites, particularly individual and small business websites that don't yet get a ton of visitors or have many files or pages on the site. But it typically means that your website doesn't have its own unique IP address. Rather, it shares one with other websites.

That generally doesn't cause issues, but there are a few situations where it can. For instance, if someone that shares your website's IP is participating in online illegal activity, your website could end up on blacklists. Blacklisted websites may become inaccessible to some visitors, and emails associated with your domain could end up in spam folders. It's not a super common issue to arise when you have a shared IP address for your website, but it's a possibility.

Dedicated IP Addresses

Some types of web hosting plans come with a dedicated IP address (or more than one) included, and in other cases you may be able to buy one as an add-on to the plan you have. Besides helping you avoid potential backlists because of bad behavior from others on your server, there are a few additional benefits to dedicated IP hosting.

For one, it gives you the option of pulling up your website using the IP address alone, rather than your domain name. This comes in handy if you want a way to start building and testing out your website before you register your domain. It also allows you to access your website while you're waiting on a domain transfer. Those are pretty niche needs, but in specific circumstances can be a benefit.

In addition, it can make using an SSL certificate for your website a little easier. At one time, a dedicated IP address was required if you wanted to use an SSL certificate.hat's luckily no longer the case. Websites with a shared IP can still invest in an SSL using Server Name Identification (SNI), as long as your web hosting service provider supports it. But having a dedicated IP address can make using an SSL certificate a little easier.

And finally, a dedicated IP address lets you run your own FTP server. That can make it easier to share and transfer files with multiple people within an organization, and provides options like allowing anonymous FTP sharing.

Versions of IP Addresses

That covers the main types of IP addresses, but it's also useful to know that there are two versions of IP addresses now available.

IPv4

IPv4 stands for internet protocol version 4. It's the most common version of IP addresses you'll see. All the IP addresses we discussed and described in this article are IPv4 addresses. Anything with the XXX.XXX.XXX.XXX construction is an IPv4 IP address.

For years IPv4 has been the main game in town, and for now it still very much is. But with the internet's rapid growth, there's concern that the over 4 billion IP addresses it could possibly generate using the IPv4 construction won't cut it. So while for practical purposes, it's the primary type of IP address most consumers will encounter, it's not the only option.

IPv6

To keep us from running out of IPv4 IP addresses, we now also have IPv6. As you'd expect, this stands for internet protocol version 6 and is set up to allow for a far greater range of addresses than its predecessor. It's also sometimes referred to as IPng, which stands for internet protocol next generation.

Where IPv4 is a 32-bit IP protocol address construction, IPv6 is 128-bit one. Instead of portions of the address being separated by periods, they're divided by colons. And addresses can include both numerical and alphabetical digits. An IPv6 address can therefore look something like this: 2001:0db8:85a3:0000:0000:8a2e:0370:7334.

That leaves room to create a *lot* more unique IP addresses for more data in the coming years, as the number of websites and devices in use continues to explode with each new configuration.

9. TCP-Host Names:

TCP/IP Host Name (hostname)

They say the best place to start is at the beginning. Therefore, in examining TCP/IP administration and troubleshooting utilities, why not begin with the basics? One of the most fundamental of tasks in diagnosing problems with a networked computer is identifying it. Just as the first thing we usually do when we meet someone is exchange names, one of the first actions an administrator takes when accessing a device is to determine its name, if it is not known. This is accomplished using the *hostname* utility.

You may recall from our discussion of TCP/IP name systems that there are two different ways that hosts can be named. The first way is to manually

assign “flat” names to devices using host tables or equivalent means; this is most often used for devices that not going to be accessed on the public Internet. The second is to give a device a domain name within the Domain Name System (DNS). The *hostname* utility can be used for both types of named hosts, but it functions in a slightly different way for each.

On most systems, including Windows and many UNIX implementations, the hostname utility is very, very simple. When the command is entered by itself on a line with no arguments, it displays the full name of the host. If it is entered with the “-s” (“short”) parameter, then if the host name is a fully qualified DNS domain name, only the local label of the node is shown and not the full domain name; if the host has a flat (non-DNS) name the parameter has no effect. A simple example is shown in Table 283.

Table 283: Using the *hostname* Utility To Check A Host Name

```
% hostname
fearn.pair.com
% hostname -s
fearn
```

The hostname utility is also intended to allow an administrator to set the name of a host. The syntax for this is also simple; you just supply the name of the host as a parameter, as follows:

hostname <new_hostname>

However, in most implementations, the use of the *hostname* command for setting a device’s name is either disabled or restricted. In Windows systems, a special applet in the Control Panel is used to set the device’s name; attempting to set it using *hostname* will result in an error message. In UNIX, the super-user of the system can use *hostname* to set the device’s name, but it is more common for this to be done by other means, such as editing the configuration file */etc/hosts*. Obviously, if a simple flat name is being assigned to this host, the administrator has full control over it, while if DNS is used then the proper procedures for registering the name must be followed.

In most operating systems, the “-s” parameter is the only one that this command supports. The parameter is not supported on all implementations of the “hostname” command, however; on some of these, if you use “hostname -s”, the system may report its host name as being “-s”. On certain Linux systems, a few additional parameters are included that allow different ways for the host name to be displayed, as well as some miscellaneous functions such as showing the version number of the program.

Note: One point worth mentioning is that the *hostname* utility is not, strictly speaking, tied into the operation of DNS or other formal mechanisms for identifying a host. It simply displays what the administrator has set it to show. Obviously it makes sense for this to be set to the host’s DNS name, but there may be exceptions.

10. FTP:

The **File Transfer Protocol** (**FTP**) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.

FTP is built on a client-server model architecture using separate control and data connections between the client and the server.[1] FTP users may authenticate themselves with a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS) or replaced with SSH File Transfer Protocol (SFTP).

The first FTP client applications were command-line programs developed before operating systems had graphical user interfaces, and are still shipped with most Windows, Unix, and Linux operating systems.[2][3] Many FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware, and FTP has been incorporated into productivity applications, such as HTML editors.

11. Telnet:

Telnet is a network protocol used to virtually access a computer and to provide a two-way, collaborative and text-based communication channel between two machines.

It follows a user command Transmission Control Protocol/Internet Protocol (TCP/IP) networking protocol for creating remote sessions. On the web, Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP) simply enable users to request specific files from remote computers, while, through Telnet, users can log on as a regular user with the privileges they are granted to the specific applications and data on that computer.

How Telnet works

Telnet is a type of client-server protocol that can be used to open a command line on a remote computer, typically a server. Users can utilize this tool to ping a port and find out whether it is open. Telnet works with what is called a *virtual terminal connection emulator*, or an abstract instance of a connection to a computer, using standard protocols to act like a physical terminal connected to a machine. FTP may also be used along with Telnet for users working to send data files.

Users connect remotely to a machine using Telnet, sometimes referred to as *Telnetting into the system*. They are prompted to enter their username and password combination to access the remote computer, which enables the running of command lines as if logged in to the computer in person. Despite the physical location of users, their IP address will match the computer logged in to rather than the one physically used to connect.

12. HTTP:

The **Hypertext Transfer Protocol** (**HTTP**) is an application layer protocol for distributed, collaborative, hypermedia information systems.[1] HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example by a mouse click or by tapping the screen in a web browser.

Development of HTTP was initiated by Tim Berners-Lee at CERN in 1989. Development of early HTTP Requests for Comments (RFCs) was a coordinated effort by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), with work later moving to the IETF.

HTTP/1.1 was first documented in RFC 2068 in 1997, and as of 2020 it (and older versions) have minority use. That specification was obsoleted by RFC 2616 in 1999, which was likewise replaced by the RFC 7230 family of RFCs in 2014.

HTTP/2 is a more efficient expression of HTTP's semantics "on the wire", and was published in 2015, has half the use of websites; it is now supported by virtually all web browsers[2] and major web servers over Transport Layer Security (TLS) using an Application-Layer Protocol Negotiation (ALPN) extension[3] where TLS 1.2 or newer is required.[4][5]

HTTP/3 is the proposed successor to HTTP/2,[6][7] which is already in use on the web, used by over 5% of desktop computers (enabled by default in latest macOS), using UDP instead of TCP for the underlying transport protocol. Like HTTP/2, it does not obsolete previous major versions of the protocol. Support for HTTP/3 was added to Cloudflare and Google Chrome in September 2019,[8][9] and can be enabled in the stable versions of Chrome and Firefox.

UNIT 2:

1 The structure of an HTML program:

Structure of an HTML Document

An HTML document has two main parts: the **head** and the **body**. But firstly every HTML document should start by declaring that it is an HTML document.

These tags are of the form:

<html>
: Should appear at the beginning of your document.

</html>
: Should appear at the end of your document.

HTML Tags

This leads us nicely on to say more about [X]HTML tags. All formatting tags (or elements) are of the general form:

<tag_on>
: Switches the tag sequence on. For example, to **bold** some text add a at the beginning of the text.

</tag_off>
: Switches the tag sequence off. The tag_on and tag_off tags are the same except the off tag has an / character in front of it. For example, to switch off the bolding add a character sequence at the end of the text that is to be given the attribute of bolding.

Note: HTML is a case insensitive mark-up language, i.e. as far as all browsers are concerned the tags <HTML> and <html> are indistinguishable. However XHTML uses only lowercase tags which will be used throughout this course.

2. Text Formatting:

Example

This text is bold

This text is italic

This is subscript and superscript

Try it Yourself »

HTML Formatting Elements

Formatting elements were designed to display special types of text:

- `<b>` - Bold text
- `<strong>` - Important text
- `<i>` - Italic text
- `<em>` - Emphasized text
- `<mark>` - Marked text
- `<small>` - Smaller text
- `<del>` - Deleted text
- `<ins>` - Inserted text
- `<sub>` - Subscript text
- `<sup>` - Superscript text

HTML <b> and Elements

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

Example

```
<b>This text is bold</b>
```

Try it Yourself »

The HTML `<strong>` element defines text with strong importance. The content inside is typically displayed in bold.

Example

```
<strong>This text is important!</strong>
```

Try it Yourself »

HTML <i> and Elements

The HTML `<i>` element defines a part of text in an alternate voice or mood. The content inside is typically displayed in italic.

Tip: The `<i>` tag is often used to indicate a technical term, a phrase from another language, a thought, a ship name, etc.

Example

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

Try it Yourself »

The HTML `<em>` element defines emphasized text. The content inside is typically displayed in italic.

Tip: A screen reader will pronounce the words in `<em>` with an emphasis, using verbal stress.

Example

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

Try it Yourself »

HTML <small> Element

The HTML `<small>` element defines smaller text:

Example

```
<small>This is some smaller text.</small>
```

Try it Yourself »

HTML <mark> Element

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

Example

```
<p>Do not forget to buy <mark>milk</mark> today.</p>
```

Try it Yourself »

HTML <del> Element

The HTML `<del>` element defines text that has been deleted from a document. Browsers will usually strike a line through deleted text:

Example

```
<p>My favorite color is <del>blue</del> red.</p>
```

Try it Yourself »

HTML <ins> Element

The HTML `<ins>` element defines a text that has been inserted into a document. Browsers will usually underline inserted text:

Example

```
<p>My favorite color is <del>blue</del> <ins>red</ins>.</p>
```

Try it Yourself »

HTML <sub> Element

The HTML `<sub>` element defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H_2O:

Example

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

Try it Yourself »

HTML <sup> Element

The HTML `<sup>` element defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font. Superscript text can be used for footnotes, like WWW[1]:

Example

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

Try it Yourself »

HTML Exercises

Exercise:

Add extra importance to the word "degradation" in the paragraph below.

```
<p>
WWF's mission is to stop the degradation of our
planet's natural environment.
</p>
```

Submit Answer »

Start the Exercise

HTML Text Formatting Elements

Tag	Description
<b>	Defines bold text
	Defines emphasized text
<i>	Defines a part of text in an alternate voice or mood
<small>	Defines smaller text
	Defines important text
<sub>	Defines subscripted text

<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<mark>	Defines marked/highlighted text

3.Text styles:

HTML Styles

The HTML `style` attribute is used to add styles to an element, such as color, font, size, and more.

Example

I am Red

I am Blue

I am Big

Try it Yourself »

The HTML Style Attribute

Setting the style of an HTML element, can be done with the `style` attribute.

The HTML `style` attribute has the following syntax:

```
<tagname style="property:value;">
```

The ***property*** is a CSS property. The ***value*** is a CSS value.

You will learn more about CSS later in this tutorial.

Background Color

The CSS `background-color` property defines the background color for an HTML element.

Example

Set the background color for a page to powderblue:

```
<body style="background-color:powderblue;">

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
```

Try it Yourself »

Example

Set background color for two different elements:

```
<body>

<h1 style="background-color:powderblue;">This is a heading</h1>
<p style="background-color:tomato;">This is a paragraph.</p>

</body>
```

Try it Yourself »

Text Color

The CSS `color` property defines the text color for an HTML element:

Example

```
<h1 style="color:blue;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>
```

Try it Yourself »

Fonts

The CSS `font-family` property defines the font to be used for an HTML element:

Example

```
<h1 style="font-family:verdana;">This is a heading</h1>
<p style="font-family:courier;">This is a paragraph.</p>
```

Try it Yourself »

Text Size

The CSS `font-size` property defines the text size for an HTML element:

Example

```
<h1 style="font-size:300%;">This is a heading</h1>
<p style="font-size:160%;">This is a paragraph.</p>
```

Try it Yourself »

Text Alignment

The CSS `text-align` property defines the horizontal text alignment for an HTML element:

Example

```
<h1 style="text-align:center;">Centered Heading</h1>
<p style="text-align:center;">Centered paragraph.</p>
```

4. Text Effects:

CSS Text Effects

CSS Text Overflow, Word Wrap, Line Breaking Rules, and Writing Modes

In this chapter you will learn about the following properties:

- `text-overflow`
- `word-wrap`
- `word-break`
- `writing-mode`

CSS Text Overflow

The CSS `text-overflow` property specifies how overflowed content that is not displayed should be signaled to the user.

It can be clipped:

This is some long text that will not fit in the box

or it can be rendered as an ellipsis (...):

This is some long text that will not fit in the box

The CSS code is as follows:

Example

```
p.test1 {
  white-space: nowrap;
  width: 200px;
  border: 1px solid #000000;
  overflow: hidden;
  text-overflow: clip;
}

p.test2 {
  white-space: nowrap;
  width: 200px;
  border: 1px solid #000000;
  overflow: hidden;
  text-overflow: ellipsis;
}
```

Try it Yourself »

The following example shows how you can display the overflowed content when hovering over the element:

Example

```
div.test:hover {
  overflow: visible;
}
```

Try it Yourself »

CSS Word Wrapping

The CSS `word-wrap` property allows long words to be able to be broken and wrap onto the next line.

If a word is too long to fit within an area, it expands outside:

This paragraph contains a very long word: thisisaveryveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

The word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word:

This paragraph contains a very long word: thisisaveryveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

The CSS code is as follows:

Example

Allow long words to be able to be broken and wrap onto the next line:

```
p {
  word-wrap: break-word;
}
```

Try it Yourself »

CSS Word Breaking

The CSS `word-break` property specifies line breaking rules.

This paragraph contains some text. This line will-break-at-hyphens.

This paragraph contains some text. The lines will break at any character.

The CSS code is as follows:

Example

```
p.test1 {
  word-break: keep-all;
}

p.test2 {
  word-break: break-all;
}
```

Try it Yourself »

CSS Writing Mode

The CSS `writing-mode` property specifies whether lines of text are laid out horizontally or vertically.

Some text with a span element with a vertical-rl writing-mode.

The following example shows some different writing modes:

Example

```
p.test1 {
  writing-mode: horizontal-tb;
}

span.test2 {
  writing-mode: vertical-rl;
}

p.test2 {
  writing-mode: vertical-rl;
```

5. **Ordered List:**

HTML Ordered Lists

An ordered list created using the `<ol>` element, and each list item starts with the `<li>` element. Ordered lists are used when the order of the list's items is important.

The list items in an ordered list are marked with numbers. Here's an example:

Example
Try this code »

```
<ol>
    <li>Fasten your seatbelt</li>
    <li>Starts the car's engine</li>
    <li>Look around and go</li>
</ol>
```

— The output of the above example will look something like this:

1. Fasten your seatbelt
2. Starts the car's engine
3. Look around and go

The numbering of items in an ordered list typically starts with 1. However, if you want to change that you can use the `start` attribute, as shown in the following example:

Example
Try this code »

```
<ol start="10">
    <li>Mix ingredients</li>
```

```
    <li>Bake in oven for an hour</li>
    <li>Allow to stand for ten minutes</li>
</ol>
```

— The output of the above example will look something like this:

10. Mix ingredients
11. Bake in oven for an hour
12. Allow to stand for ten minutes

Like unordered list, you can also use the CSS `list-style-type` property to change the numbering type in an ordered list. The following style rule changes the marker type to roman numbers.

Example
Try this code »

```
ol {
    list-style-type: upper-roman;
}
```

6. Unordered List:

The HTML `<ul>` tag defines an unordered (bulleted) list.

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>
```

Try it Yourself »

Unordered HTML List - Choose List Item Marker

The CSS `list-style-type` property is used to define the style of the list item marker. It can have one of the following values:

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>
```

Try it Yourself »

Example - Circle

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

Try it Yourself »

Example - Square

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

Try it Yourself »

Example - None

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

Try it Yourself »

Nested HTML Lists

Lists can be nested (list inside list):

Example

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

Try it Yourself »

Note: A list item (<li>) can contain a new list, and other HTML elements, like images and links, etc.

Horizontal List with CSS

HTML lists can be styled in many different ways with CSS.

One popular way is to style a list horizontally, to create a navigation menu:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333333;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 16px;
  text-decoration: none;
}

li a:hover {
  background-color: #111111;
}
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

</body>
</html>
```

Try it Yourself »

Tip: You can learn much more about CSS in our CSS Tutorial.

Chapter Summary

- Use the HTML `<ul>` element to define an unordered list
- Use the CSS `list-style-type` property to define the list item marker
- Use the HTML `<li>` element to define a list item
- Lists can be nested
- List items can contain other HTML elements
- Use the CSS property `float:left` to display a list horizontally

HTML List Tags

Tag	Description
	Defines an unordered list
	Defines an ordered list
	Defines a list item
<dl>	Defines a description list
<dt>	Defines a term in a description list
<dd>	Describes the term in a description list

7. **Adding graphics to HTML documents:**

Adding graphics to HTML files

There are two ways to include images (graphics) in an HTML document: inline and external. You'll usually use inline images, which appear directly in the HTML page. External images are downloaded when a user clicks a link to the image.

Some browsers can read a wide variety of types of image files, but some can read only a few. The most commonly readable format is `.GIF` files. There are lots of shareware products that create `.GIFs` or translate one type of image (for example `.BMP`) to `.GIF`. `JPEG` image files are usually better for photographic images, or images with a lot of detail. GIF images can be better for images with large expanses of solid color.

To include an image in your HTML document, use the <IMG> tag.

<IMG SRC="some.gif">

The previous line includes the file some.gif in your HTML document. This assumes that the file is in the same directory as your HTML document. If the file is in another directory, use either the relative or absolute path.

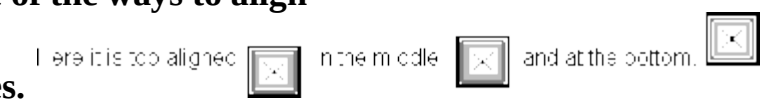
You can include images on separate lines, or you can include them in text in headings, body paragraphs, and even lists.

Elements of <IMG>

The image tag has several attributes that control the graphic. The first is SRC. This defines the source for the graphic--the `.GIF` image file.

You can control where the image is positioned relative to the text of the line it appears in by using the ALIGN attribute. You can set ALIGN to one of nine different values: LEFT, RIGHT, TOP, ABSMIDDLE, ABSBOTTOM, TEXTTOP, MIDDLE, BASELINE, or BOTTOM. If you don't specify alignment, it defaults to BOTTOM.

Three of the ways to align images.



Some browsers can't display images. You can include a text string that describes the image by using the ALT attribute.

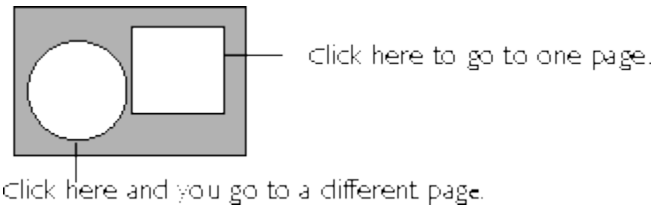
The following example displays an image and includes the descriptive text for browsers that can't display images:

```
<IMG SRC="icon.gif" ALT="This is my icon">
```

The following example shows a series of text blocks with images aligned according to many of the alignment options offered by Navigator:

```
<hr>

text around <img src=red_ball.gif width=18 height=18> the image.

<hr>

<b>top</b> text around

<img src=red_ball.gif align=top width=18 height=18>

the image.

<hr>

<b>absmiddle</b> text around

<img src=red_ball.gif align=absmiddle width=18 height=18>

the image.

<hr>

<b>absbottom</b> text around

<img src=red_ball.gif align=absbottom width=18 height=18>

the image.

<hr>

<b>baseline</b> text around

<img src=red_ball.gif align=baseline width=18 height=18>

the image.

<hr>

<b>left</b> text around

<img src=red_ball.gif align=left width=18 height=18>

the image.
```

```
Oh, if only we could make the text flow around an image.  We could do
such great formatting if we had the capability to let text flow around
images of all kinds: photos of animals, plants, buildings, and people,
or simple line drawings of various helpful sorts.
<br clear=all>
<hr>
<b>right</b> text around
<img src=red_ball.gif  align=right  width=18 height=18></applet>
the image.
Oh, if only we could make the text flow around an image.  We could do
such great formatting if we had the capability to let text flow around
images of all kinds: photos of animals, plants, buildings, and people,
or simple line drawings of various helpful sorts.
<br clear=all>
<hr>
```



Linking images to other pages

You can use graphics as links to other pages by embedding the image tag in a link. The following example adds a circle graphic and links it to the HTML document called circles.

```
<A HREF="circles.html"><IMG SRC="circle.gif"></A>
```

You can combine graphics and text in one link. This means that you can click either the graphic or the text to jump to the corresponding page:

```
<A HREF="icons.html"><IMG SRC="myicon.gif">My icon is cool</A>
```

Interlaced GIFs

Interlacing is a feature of the GIF file format that lets a viewer or browser display a file with progressively greater detail in four successive passes. When Navigator loads an interlaced GIF, the first pass starts at row 0 and displays every eighth row of bitmap data; the second pass starts at row 4 and also displays every eighth row. Similarly, the third and fourth passes load progressively more data in a non-linear fashion until the full image is loaded.

When Navigator loads an interlaced GIF, the image appears to gradually fade in. When 50% of an interlaced GIF is loaded, a user can interpret the image because every fourth row of data is visible. When Navigator loads a non-interlaced GIF, each row of data is loaded sequentially. When 50% of a non-interlaced GIF is loaded, only the top half of the image is visible.

Navigator and most other browsers that support the GIF format support interlaced GIFs. Third-party tools are available on the Internet to let you convert a non-interlaced GIF to the interlaced format.

Image maps

An image map is a graphic that has clickable regions that link to different pages. For example, you can have an image with a square and a circle where a click in the square takes you to one page and a click in the circle takes you to a different page.

Different areas of an image map can jump to different Web pages.

HTML provides two forms of image maps:

- Client-side image maps are defined by the combination of MAP and AREA tags and are loaded into the Navigator as hypergraphics. When a user clicks the image, Navigator determines what URL to load based on the information in the AREA tag. The USEMAP attribute of the IMG tag specifies an image as a client-side image map. Client-side image maps usually respond quickly, and can be developed and tested locally.
- Server-side image maps are defined by the combination of A and IMG tags and are loaded into the Navigator as ordinary IMG files. When a user clicks the image, the server determines what URL to load. The ISMAP attribute of the MG tag specifies an image as a server-side image map. Server-side image maps

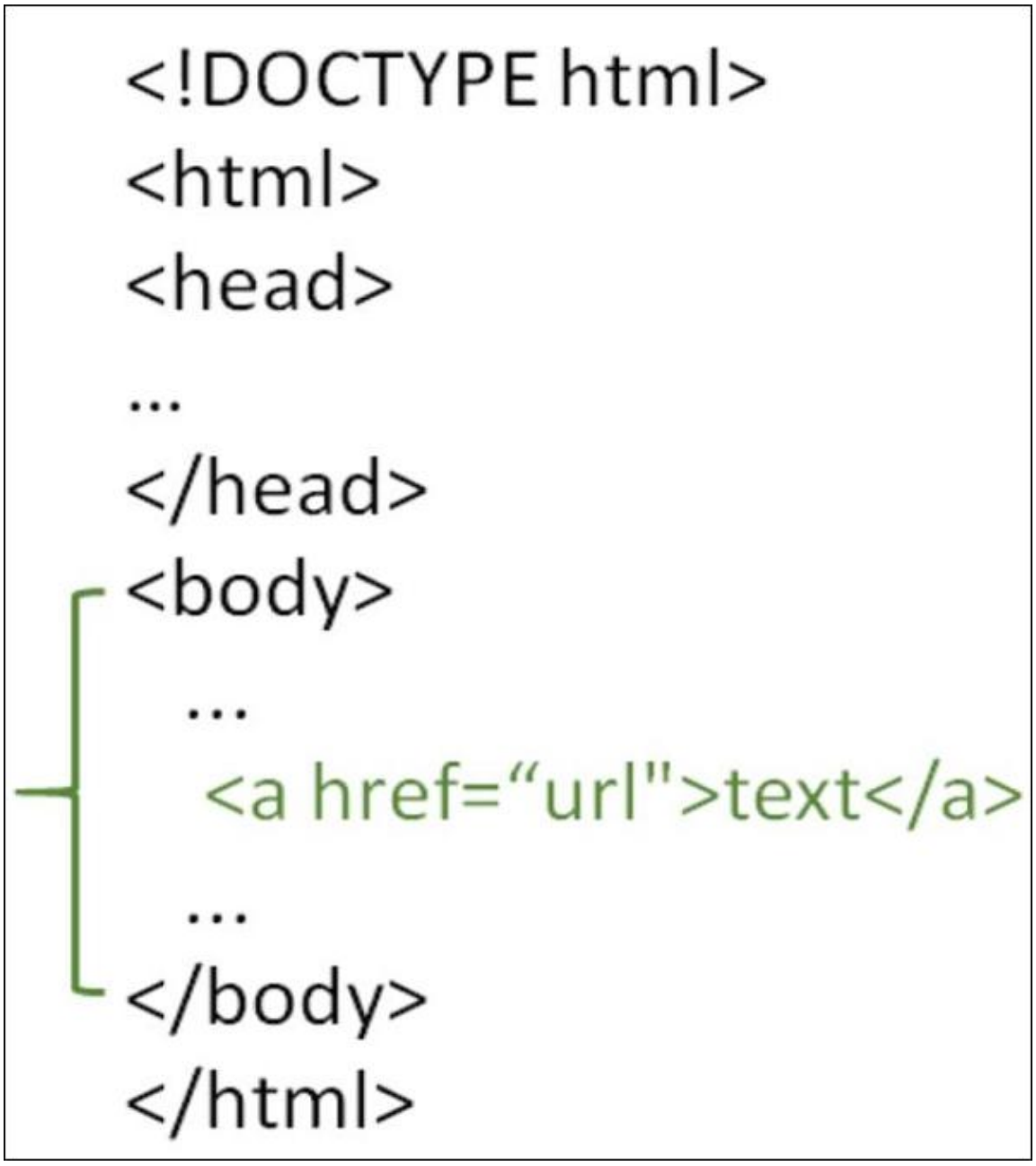
can be slower to respond, because the processing usually takes place on the server. For information on how to use Navigator JavaScript to do the processing locally, see the JavaScript Guide.

Unit III

1. **Linking of Web Pages:**

With HTML, easily add page links to an HTML page. Link contact us, about, home or any other external website page using the page links, which gets added inside an HTML document. To make page links in an HTML page, use the <a> and tags, which are the tags used to define the links.

The <a> tag indicates where the link starts and the tag indicates where it ends. Whatever text gets added inside these tags, will work as a link. Add the URL for the link in the . Just keep in mind that you should use the <a>… tags inside <body>…</body> tags.



Example

You can try the following code to make page links in an HTML page:

Live Demo

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Links</title>
```

```
    </head>


    <body>
        <h1>Reach us here</h1>
        <a href="/about/contact_us.htm">Contact</a>
    </body>
</html>
```

2. **- Internal Links:**

HTML Links - HTML Internal Link

- *« Previous*
- *Next »*

HTML internal link is linked within the same web page. This link can be an absolute path or relative path.

HTML internal link name is followed by the hash sign(#). You have to assign an `id` to refer section of your page, which is referred to as an internal link to the same page.

When you click on an internal anchor link, you will scroll automatically to the referred section and display it on your browser.

To understand internal link see the below examples.

- *Lession.1* link can be referred as *Introduction of Lession.1* automatically.
- *Lession.2* link can be referred as * Introduction of Lession.2 * automatically.

> **Notes**: You can not use `name` attribute instead of `id` attribute. Because `name` attribute not supported in HTML5. Use the `id` attribute instead of.

Example

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
  <a href="#lession1">Lession.1</a><br />
  <a href="#lession2">Lession.2</a><br />
  <a href="#lession3">Lession.3</a><br />
  <a href="#lession4">Lession.4</a><br />
  <br />

  <a id="lession1">Introduction of Lession.1</a>
  <p>This is sub topic.1</p>
  <p>This is sub topic.2</p>
  <p>This is sub topic.3</p>
  <p>This is sub topic.4</p>
  <br />
  <br />
  <div id="lession2">Introduction of Lession.2</div>
  <p>This is sub topic.1</p>
  <p>This is sub topic.2</p>
  <p>This is sub topic.3</p>
  <p>This is sub topic.4</p>
  <br />
  <br />
  <p id="lession3">Introduction of Lession.3</p>
  <p>This is sub topic.1</p>
  <p>This is sub topic.2</p>
  <p>This is sub topic.3</p>
  <p>This is sub topic.4</p>
  <br />
  <br />
  <article id="lession4">Introduction of Lession.4</article>
  <p>This is sub topic.1</p>
```

```
  <p>This is sub topic.2</p>
  <p>This is sub topic.3</p>
  <p>This is sub topic.4</p>
</body>
</html>
```

Run it... »

Linking to parts of other documents

You can set target to specific sections of the other pages by adding the #id at the end of the href. After adding hash mark is known as a "fragment identifier". This helps a lot for example, you can link to the third section of this tutorial from anywhere else, you have to write

```
<a href="https://way2tutorial.com/html/html_internal_links.php#section-id">
```

3. Tables:

Define an HTML Table

The <table> tag defines an HTML table.

Each table row is defined with a <tr> tag. Each table header is defined with a <th> tag. Each table data/cell is defined with a <td> tag.

By default, the text in <th> elements are bold and centered.

By default, the text in <td> elements are regular and left-aligned.

Example

A simple HTML table:

```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
```

```
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Try it Yourself »

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 - Add a Border

To add a border to a table, use the CSS border property:

Example

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

Try it Yourself »

Remember to define borders for both the table and the table cells.

HTML Table - Collapsed Borders

To let the borders collapse into one border, add the CSS border-collapse property:

Example

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

Try it Yourself »

HTML Table - Add Cell Padding

Cell padding specifies the space between the cell content and its borders.

If you do not specify a padding, the table cells will be displayed without padding.

To set the padding, use the CSS `padding` property:

Example

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

Try it Yourself »

HTML Table - Left-align Headings

By default, table headings are bold and centered.

To left-align the table headings, use the CSS `text-align` property:

Example

```
th {
  text-align: left;
}
```

Try it Yourself »

HTML Table - Add Border Spacing

Border spacing specifies the space between the cells.

To set the border spacing for a table, use the CSS `border-spacing` property:

Example

```
table {
  border-spacing: 5px;
}
```
Try it Yourself »

Note: If the table has collapsed borders, `border-spacing` has no effect.

HTML Table - Cell 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>Bill Gates</td>
    <td>55577854</td>
    <td>55577855</td>
  </tr>
</table>
```
Try it Yourself »

HTML Table - Cell 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>Bill Gates</td>
  </tr>
  <tr>
    <th rowspan="2">Telephone:</th>
    <td>55577854</td>
  </tr>
  <tr>
    <td>55577855</td>
```

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

Try it Yourself »

HTML Table - Add 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>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$50</td>
  </tr>
</table>
```

HTML Table Tags

Tag	Description
<table>	Defines a table
<th>	Defines a header cell in a table
<tr>	Defines a row in a table

<td>	Defines a cell in a table
<caption>	Defines a table caption
<colgroup>	Specifies a group of one or more columns in a table for formatting
<col>	Specifies column properties for each column within a <colgroup> element
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table

For a complete list of all available HTML tags, visit our HTML Tag

4. Frames:

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

Disadvantages of Frames

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages −

- Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.

- Sometimes your 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 that do not support frame technology.

Creating Frames

To use frames on a page we use <frameset> tag instead of <body> tag. The <frameset> tag defines, how to divide the window into frames. The **rows** attribute of <frameset> tag defines horizontal frames and **cols** attribute defines vertical frames. Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

Note − The <frame> tag deprecated in HTML5. Do not use this element.

Example

Following is the example to create three horizontal frames −

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Frames</title>
   </head>

   <frameset rows = "10%,80%,10%">
      <frame name = "top" src = "/html/top_frame.htm" />
      <frame name = "main" src = "/html/main_frame.htm" />
      <frame name = "bottom" src = "/html/bottom_frame.htm" />

      <noframes>
         <body>Your browser does not support frames.</body>
      </noframes>

   </frameset>

</html>
```

This will produce the following result −

Example

Let's put the above example as follows, here we replaced rows attribute by cols and changed their width. This will create all the three frames vertically −

```
<!DOCTYPE html>
<html>

   <head>
```

```
      <title>HTML Frames</title>
   </head>

   <frameset cols = "25%,50%,25%">
      <frame name = "left" src = "/html/top_frame.htm" />
      <frame name = "center" src = "/html/main_frame.htm" />
      <frame name = "right" src = "/html/bottom_frame.htm" />

      <noframes>
         <body>Your browser does not support frames.</body>
      </noframes>
   </frameset>

</html>
```

This will produce the following result −

The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag −

Sr.No	Attribute & Description
1	**cols** Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways − Absolute values in pixels. For example, to create three vertical frames, use *cols = "100, 500, 100"*. A percentage of the browser window. For example, to create three vertical frames, use *cols = "10%, 80%, 10%"*. Using a wildcard symbol. For example, to create three vertical frames, use *cols = "10%, *, 10%"*. In this case wildcard takes remainder of the window. As relative widths of the browser window. For example, to create three vertical frames, use *cols = "3*, 2*, 1*"*. This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.
2	**rows** This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use *rows = "10%, 90%"*. You can specify the height of each row in the same way as explained above for columns.

3	**border** This attribute specifies the width of the border of each frame in pixels. For example, border = "5". A value of zero means no border.
4	**frameborder** This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder = "0" specifies no border.
5	**framespacing** This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing = "10" means there should be 10 pixels spacing between each frames.

The <frame> Tag Attributes

Following are the important attributes of <frame> tag −

Sr.No	Attribute & Description
1	**src** This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top_frame.htm" will load an HTML file available in html directory.
2	**name** This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
3	**frameborder** This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
4	**marginwidth**

	This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth = "10".
5	**marginheight** This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight = "10".
6	**noresize** By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize = "noresize".
7	**scrolling** This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars.
8	**longdesc** This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc = "framedescription.htm"

5.CSS:

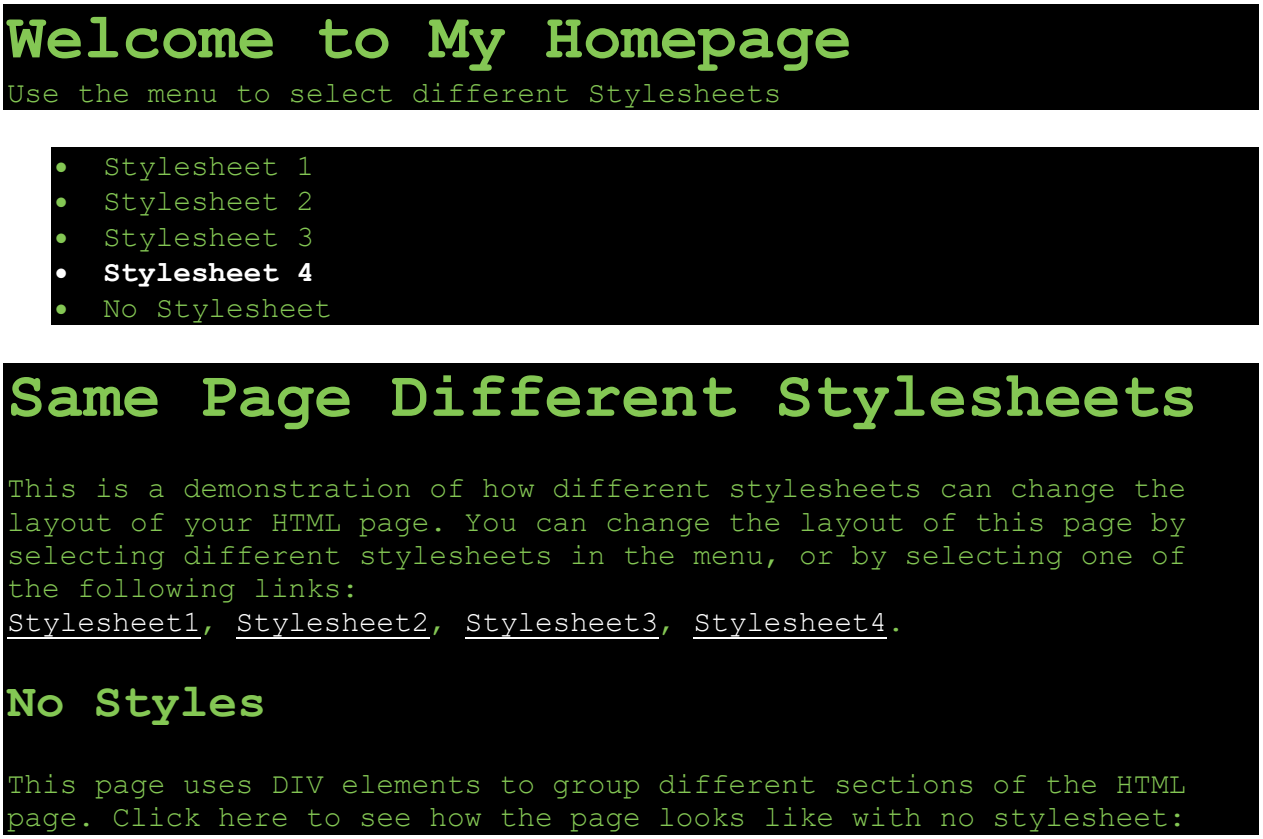
CSS Introduction

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
- External stylesheets are stored in **CSS files**

CSS Demo - One HTML Page - Multiple Styles!

Here we will show one HTML page displayed with four different stylesheets. Click on the "Stylesheet 1", "Stylesheet 2", "Stylesheet 3", "Stylesheet 4" links below to see the different styles:



6. Font attributes:

Font Attribute

Specifies a compound value of font attributes. Read/write **String**.
Applies To
TextPath
Tag Syntax
<v: *element* **style="font:***expression***">**
Script Syntax
*element***.style.font**="*expression*"
*expression=element***.style.font**
Remarks
Defines the attributes of a font, including family, style, weight, size, and variant. The order of definitions in the string is: **font-style**, **font-variant**, **font-weight**, **font-size**, **line-height**, and **font-family**. The values are the same as the standard HTML style attributes.
VML Standard Attribute
Example
The font of the textpath text is italic, small-caps, bold, 36 point Arial.

```
<v:line from="50 100" to="400 100">

<v:fill on="True" color="red"/>

<v:path textpathok="True"/>

<v:textpath on="True" string="VML Text"

style="font:italic small-caps bold 36pt Arial"/>

</v:line>
```

7.Color and Background attributes:

HTML | bgcolor attribute

Last Updated: 30-09-2019

The **HTML bgcolor attribute** is used to set the background color of an HTML element. Bgcolor is one of those attributes that has become deprecated with the implementation of Cascading Style Sheets (see CSS Backgrounds).

Syntax:

```
<"tag" bgcolor="Value">
```

Supported tags:

- body
- marquee
- table
- tBody
- td
- tFoot
- th
- tHead
- tr

Example: HTML <table> bgcolor attribute

filter_none

edit

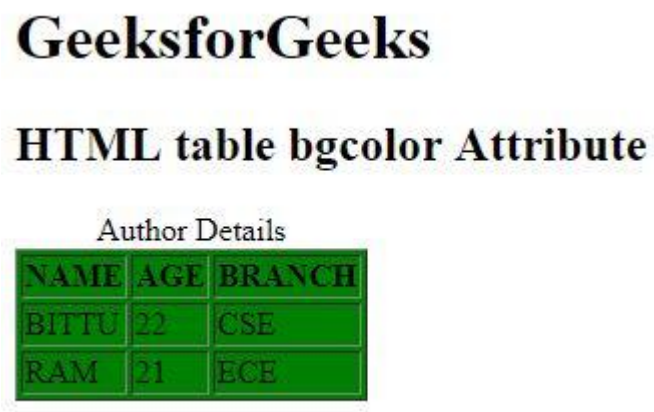
play_arrow

brightness_4

```
<!DOCTYPE html>
<html>

<head>
    <title>
        HTML table bgcolor Attribute
    </title>
</head>

<body>
    <h1>GeeksforGeeks</h1>

    <h2>HTML table bgcolor Attribute</h2>

    <table border="1"
        bgcolor="green">
        <caption>
        Author Details
    </caption>

        <tr>
            <th>NAME</th>
```

```
            <th>AGE</th>
            <th>BRANCH</th>
        </tr>
        <tr>
            <td>BITTU</td>
            <td>22</td>
            <td>CSE</td>
        </tr>
        <tr>
            <td>RAM</td>
            <td>21</td>
            <td>ECE</td>
        </tr>
    </table>
</body>

</html>
```

Output:



Example:HTML body Bgcolor Attribute

filter_none
edit
play_arrow
brightness_4

```
<!DOCTYPE html>
<html>

<head>
    <title>
    HTML body Bgcolor Attribute
</title>
</head>

<!-- body tag starts here -->

<body text="green" bgcolor="orange">
```

```
    <center>
        <h1>GeeksforGeeks</h1>
        <h2>
        HTML <body> bgcolor Attribute
    </h2>
        <p>
        It is a Computer
        Science portal For Geeks
    </p>
    </center>
</body>
<!-- body tag ends here -->

</html>
```

Output:

Note: The bgcolor attribute is not supported in HTML5.
Supported Browsers: The browsers supported by **bgcolor attribute** are listed below:

- Google Chrome
- Internet Explorer
- Firefox
- Apple Safari
- Opera

8. Text attributes:

HTML | body text Attribute

Last Updated: 07-08-2019
The **HTML <body> text Attribute** is used to define a color for the text in the Document.
Syntax:

```
<body text="color_name | hex_number | rgb_number">
```

Attribute Values

- **color_name:** It specify the name of the color for the text in the Document.
- **hex_number:** It specify the hex code of the color of the Text in the Document.

- **rgb_number:** It specify the rgb value of the Text in the Document

Example:

filter_none
edit
play_arrow
brightness_4

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML body Text Attribute</title>
</head>

<!-- body tag starts here -->

<body text="green">
    <center>
        <h1>GeeksforGeeks</h1>
        <h2>HTML <body> Text Attribute</h2>
        <p>It is a Computer Science portal For Geeks</p>
    </center>
</body>
<!-- body tag ends here -->

</html>
```

Output:

Supported Browsers: The browser supported by **<body> Text** are listed below:

- Google Chrome
- Internet Explorer
- Firefox
- Safari
- Opera

9. Border attributes:

HTML | border attribute

Last Updated: 07-11-2020
The **HTML border attribute** is used to set visible border width to most HTML elements within the body.
Syntax:

```
<tag border="value">
```

Supported tags:

- Table
- Image
- Object

Example: table border attribute.

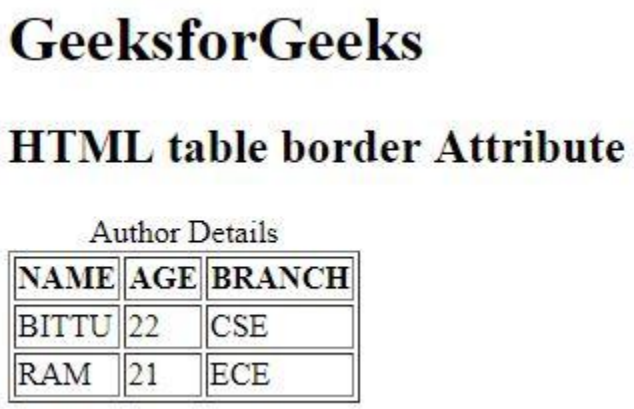
HTML

filter_none
edit
play_arrow
brightness_4

```
<!DOCTYPE html>
<html>

<head>
    <title>
        HTML table border Attribute
    </title>
</head>

<body>
    <h1>GeeksforGeeks</h1>

    <h2>HTML table border Attribute</h2>

    <table border="1">
        <caption>Author Details</caption>

        <tr>
            <th>NAME</th>
            <th>AGE</th>
            <th>BRANCH</th>
        </tr>
        <tr>
            <td>BITTU</td>
            <td>22</td>
            <td>CSE</td>
        </tr>
```

```
        <tr>
            <td>RAM</td>
            <td>21</td>
            <td>ECE</td>
        </tr>
    </table>
</body>

</html>
```

Output:



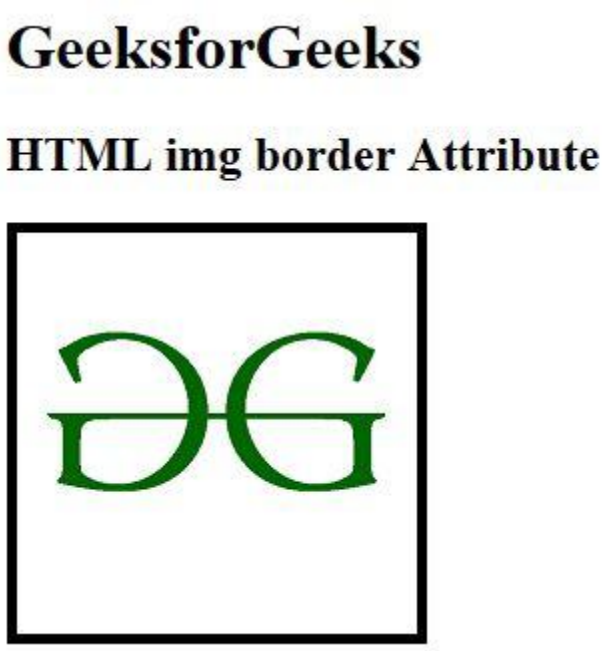
Example: Img border attribute.

HTML

filter_none
edit
play_arrow
brightness_4

```
<!DOCTYPE html>
<html>

<head>
    <title>
        HTML img border Attribute
    </title>
</head>

<body>
    <h1>GeeksforGeeks</h1>

    <h2>HTML img border Attribute</h2>

    <img src=
```

"https://media.geeksforgeeks.org/wp-content/uploads/20190506164011/logo3.png"
```
        alt="GeeksforGeeks logo"
        border="5">
</body>

</html>
```
Output:



Example: Object border attribute.

HTML

filter_none
edit
play_arrow
brightness_4

```
<!DOCTYPE html>
<html>

<head>
    <title>
        HTML object border Attribute
    </title>
</head>

<body>
    <center>
        <h1>GeeksForGeeks</h1>
```

```
        <h1>
            HTML <object> border Attribute
        </h1>
        <br>

        <object data=
"https://www.geeksforgeeks.org/wp-content/uploads/Geek_logi_-low_res.png"
                width="550px" height="150px" border="4">
            GeeksforGeeks
        </object>
    </center>
</body>

</html>
```

Output:

Supported Browsers: The browsers supported by **HTML border attribute** are listed below:

- Google Chrome
- Internet Explorer
- Firefox
- Apple Safari
- Opera

Unit IV

1. Advantages of JavaScript

JavaScript is one of the easiest, versatile and effective languages used to extend functionality in websites. **JavaScript Development Services** helps in on-screen visual effects and processing and calculating data on web pages with ease. The programming language also helps in extended functionality to websites using third party scripts among several other handy features.

In this blog, we are going the discuss some of the key features of JavaScript.

Here are some key Advantages of JavaScript:

1. JavaScript is a client side language

The JavaScript code is executed on the user's processor instead of the web server thus it saves bandwidth and load on the web server.

2. JavaScript is an easy language to learn

The JavaScript language is easy to learn and offers syntax that is close to English. It uses the DOM model that provides plenty of predefined functionalities to the various objects on pages making it a breeze to develop a script to solve a custom purpose.

3. JavaScript is comparatively fast for the end user

As the code is executed on the client side, results and processing is completed almost instantly depending on the task (tasks in JavaScript on web pages are usually simple so as to prevent being a memory hog) as it does not need to be processed in the site's web server and sent back to the user consuming local as well as server bandwidth.

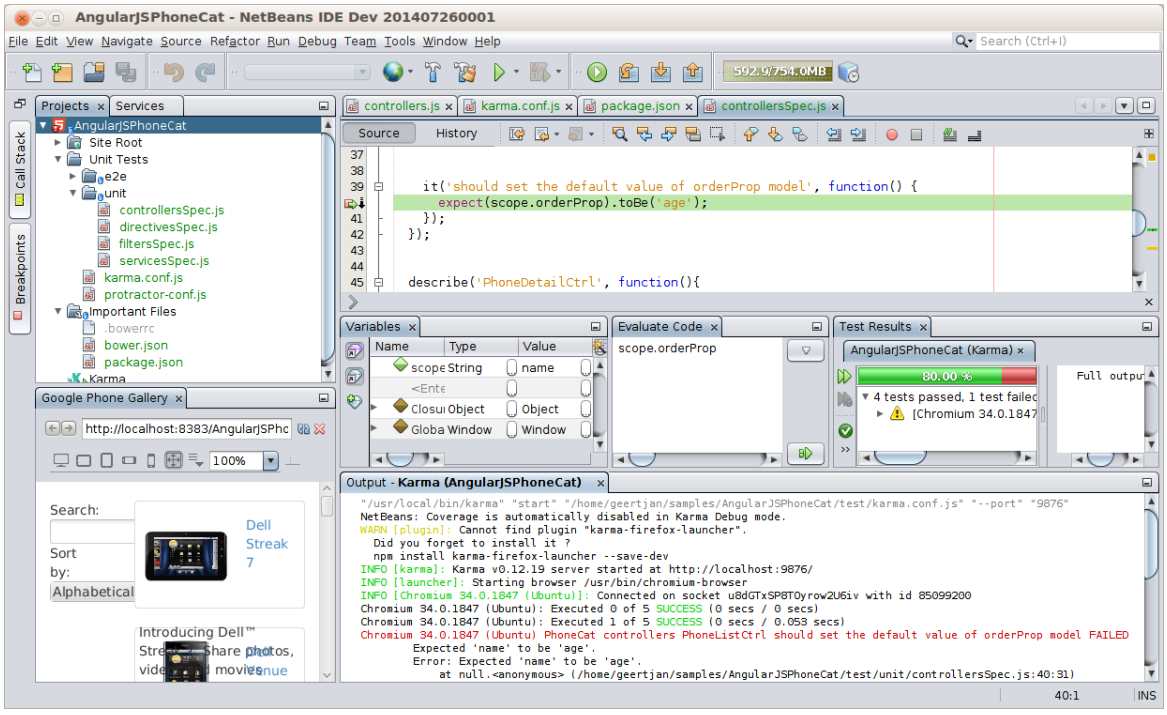
4. Extended functionality to web pages

Third party add-ons like Greasemonkey enable JavaScript developers to write snippets of JavaScript which can execute on desired web pages to extend its functionality. If you use a website and require a certain feature to be included, you can write it by yourself and use an add-on like Greasemonkey to implement it on the web page.

5. No compilation needed

JavaScript does not require compilation process so no compiler is needed. The browser interprets JavaScript as it HTML tags.

6. Easy to debug and test



The understanding syntax of JavaScript is easy. Any person can learn it very easily and use it to develop dynamic and scalable websites.

7. Platform independent

Any JavaScript-enabled browser can understand and interpret JavaScript code. Any JavaScript code can be executed on different types of hardware a JavaScript program written for.

8. Event-Based Programming language

Being an event-based language, different code segments are executed whenever a certain event occurs in JavaScript. In simple language, a code segment is executed when a user clicks a button or moves a mouse over the object.

Related Blog: Top 5 JavaScript Frameworks in 2017

9. Procedural programming capabilities

JavaScript language encompasses all the capabilities of a procedural language. Branching, looping, condition checking are some of those capabilities that can be executed on a web page.

This infographic explains why JavaScript is the programming language of the future:

Conclusion:

JavaScript has become a vital need for web development in 2017. However, we have listed out top 9 advantages of JavaScript and an infographic is also attached which can be helpful for your JavaScript development project.

If you have any query related to JavaScript, you can **Contact Markupbox,** leading JavaScript development company in India which provides expert dedicated developers for all kinds of business projects.

2. Java script into HTML:

How To Add JavaScript to HTML

JavaScript

Introduction

JavaScript, also abbreviated to JS, is a programming language used in web development. As one of the core technologies of the web alongside HTML and CSS, JavaScript is used to make webpages interactive and to build web apps. Modern web browsers, which adhere to common display standards, support JavaScript through built-in engines without the need for additional plugins.

When working with files for the web, JavaScript needs to be loaded and run alongside HTML markup. This can be done either inline within an HTML document or in a separate file that the browser will download alongside the HTML document.

This tutorial will go over how to incorporate JavaScript into your web files, both inline into an HTML document and as a separate file.

Adding JavaScript into an HTML Document

You can add JavaScript code in an HTML document by employing the dedicated HTML tag `<script>` that wraps around JavaScript code.

The `<script>` tag can be placed in the `<head>` section of your HTML, in the `<body>` section, or after the `</body>` close tag, depending on when you want the JavaScript to load.

Generally, JavaScript code can go inside of the document `<head>` section in order to keep them contained and out of the main content of your HTML document.

However, if your script needs to run at a certain point within a page's layout — like when using `document.write` to generate content — you should put it at the point where it should be called, usually within the `<body>` section.

Let's consider the following blank HTML document with a browser title of `Today's Date`:

index.html

```
<!DOCTYPE html>
<html lang="en-US">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Today's Date</title>
</head>

<body>


</body>


</html>
```

Copy

Right now, this file only contains HTML markup. Let's say we would like to add the following JavaScript code to the document:

```
let d = new Date();
alert("Today's date is " + d);
```

Copy

This will enable the webpage to display an alert with the current date regardless of when the user loads the site.

In order to achieve this, we will add a `<script>` tag along with some JavaScript code into the HTML file.

To begin with, we'll add the JavaScript code between the `<head>` tags, signalling the browser to run the JavaScript script before loading in the rest of the page. We can add the JavaScript below the `<title>` tags, for instance, as shown below:

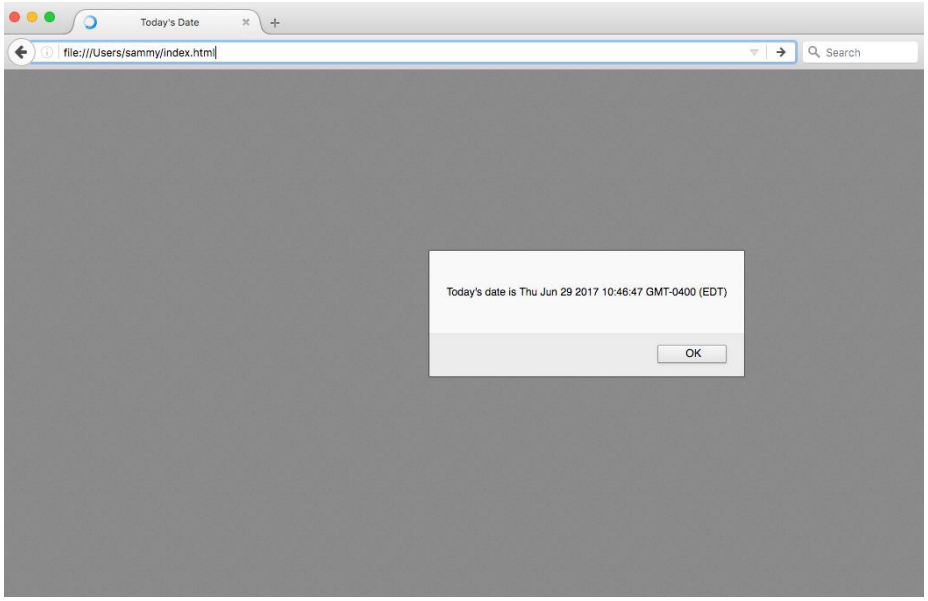
index.html

```
<!DOCTYPE html>
<html lang="en-US">


<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Today's Date</title>
    <script>
        let d = new Date();
        alert("Today's date is " + d);
```

```
    </script>
</head>

<body>

</body>



</html>
```

Copy

Once you load the page, you will receive an alert that will look similar to this:



You can also experiment with putting the script either inside or outside the `<body>` tags and reload the page. As this is not a robust HTML document, you likely will not notice any difference in the loading of the page.

If we were modifying what is shown in the body of the HTML, we would need to implement that after the `<head>` section so that it displays on the page, as in the example below:

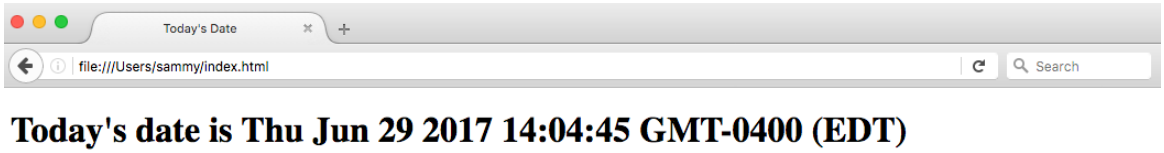
index.html

```
<!DOCTYPE html>
<html lang="en-US">


<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Today's Date</title>
</head>

<body>

  <script>
      let d = new Date();
      document.body.innerHTML = "<h1>Today's date is " + d + "</h1>"
  </script>

</body>

</html>
```

Copy

The output for the above HTML document loaded through a web browser would look similar to the following:



Today's date is Thu Jun 29 2017 14:04:45 GMT-0400 (EDT)

Scripts that are small or that run only on one page can work fine within an HTML file, but for larger scripts or scripts that will be used on many pages, it is not a very effective solution because including it can become unwieldy or difficult to read and understand. In the next section, we'll go over how to handle a separate JavaScript file in your HTML document.

Working with a Separate JavaScript File

In order to accommodate larger scripts or scripts that will be used across several pages, JavaScript code generally lives in one or more `js` files that are referenced within HTML documents, similarly to how external assets like CSS are referenced.

The benefits of using a separate JavaScript file include:

- Separating the HTML markup and JavaScript code to make both more straightforward
- Separate files makes maintenance easier
- When JavaScript files are cached, pages load more quickly

To demonstrate how to connect a JavaScript document to an HTML document, let's create a small web project. It will consist of `script.js` in the `js/` directory, `style.css` in the `css/` directory, and a main `index.html` in the root of the project.

```
project/
├── css/
|   └── style.css
├── js/
|   └── script.js
└── index.html
```

We can start with our previous HTML template from the section above:

index.html

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Today's Date</title>
</head>

<body>


</body>

</html>
```

Copy

Now, let's move our JavaScript code that will show the date as an `<h1>` header to the `script.js` file:

script.js

```
let d = new Date();
document.body.innerHTML = "<h1>Today's date is " + d + "</h1>"
```

Copy

We can add a reference to this script to or below the `<body>` section, with the following line of code:

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

Copy

The `<script>` tag is pointing to the `script.js` file in the `js/` directory of our web project.

Let's look at this line in the context of our HTML file, in this case, below the `<body>` section:

index.html

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Today's Date</title>
</head>

<body>

</body>

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

</html>
```

Copy

Finally, let's also edit the `style.css` file by adding a background color and style to the `<h1>` header:

style.css

```
body {
    background-color: #0080ff;
```

```
}

h1 {
    color: #fff;
    font-family: Arial, Helvetica, sans-serif;
}
```

Copy

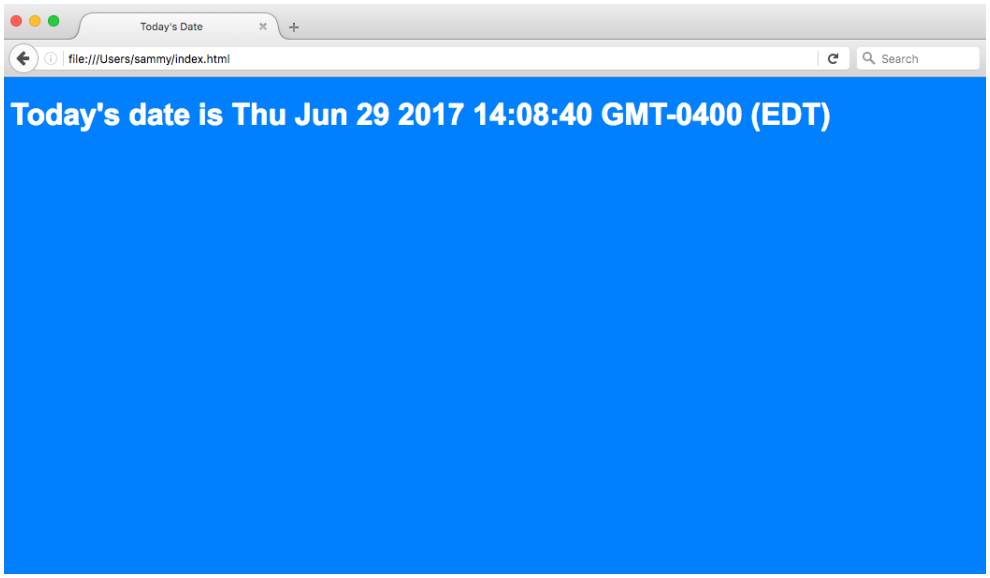
We can reference that CSS file within the `<head>` section of our HTML document:

index.html

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Today's Date</title>
    <link rel="stylesheet" href="css/style.css">
</head>

<body>

</body>

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

</html>
```

Copy

Now, with the JavaScript and CSS in place we can load the `index.html` page into the web browser of our choice. We should see a page that looks similar to the following:



Now that we've placed the JavaScript in a file, we can call it in the same way from additional web pages and update them all in a single location

Conclusion

This tutorial went over how to incorporate JavaScript into your web files, both inline into an HTML document and as a separate `.js` file.

From here, you can learn how to work with the JavaScript Developer Console and how to write comments in JavaScript.

3. Java script syntax-variables. Operators:

JavaScript Syntax

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

```
var x, y, z;        // Declare Variables
x = 5; y = 6;       // Assign Values
z = x + y;          // Compute Values
```

JavaScript Values

The JavaScript syntax defines two types of values:

- Fixed values
- Variable values

Fixed values are called **Literals**.

Variable values are called **Variables**.

JavaScript Literals

The two most important syntax rules for fixed values are:

1. **Numbers** are written with or without decimals:

```
10.50

1001
```

Try it Yourself »

2. **Strings** are text, written within double or single quotes:

```
"John Doe"

'John Doe'
```

Try it Yourself »

JavaScript Variables

In a programming language, **variables** are used to **store** data values.

JavaScript uses the var keyword to **declare** variables.

An **equal sign** is used to **assign values** to variables.

In this example, x is defined as a variable. Then, x is assigned (given) the value 6:

```
var x;

x = 6;
```

Try it Yourself »

JavaScript Operators

JavaScript uses **arithmetic operators** (+ - * /) to **compute** values:

```
(5 + 6) * 10
```

Try it Yourself »

JavaScript uses an **assignment operator** (=) to **assign** values to variables:

```
var x, y;
x = 5;
y = 6;
```

Try it Yourself »

JavaScript Expressions

An expression is a combination of values, variables, and operators, which computes to a value.

The computation is called an evaluation.

For example, 5 * 10 evaluates to 50:

```
5 * 10
```

Try it Yourself »

Expressions can also contain variable values:

```
x * 10
```

Try it Yourself »

The values can be of various types, such as numbers and strings.

For example, "John" + " " + "Doe", evaluates to "John Doe":

```
"John" + " " + "Doe"
```

Try it Yourself »

JavaScript Keywords

JavaScript **keywords** are used to identify actions to be performed.

The var keyword tells the browser to create variables:

```
var x, y;
x = 5 + 6;
y = x * 10;
```

Try it Yourself »

JavaScript Comments

Not all JavaScript statements are "executed".

Code after double slashes // or between /* and */ is treated as a **comment**.

Comments are ignored, and will not be executed:

```
var x = 5;   // I will be executed

// var x = 6;   I will NOT be executed
```

Try it Yourself »

You will learn more about comments in a later chapter.

JavaScript Identifiers

Identifiers are names.

In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).

The rules for legal names are much the same in most programming languages.

In JavaScript, the first character must be a letter, or an underscore (_), or a dollar sign ($).

Subsequent characters may be letters, digits, underscores, or dollar signs.

Numbers are not allowed as the first character.
This way JavaScript can easily distinguish identifiers from numbers.

JavaScript is Case Sensitive

All JavaScript identifiers are **case sensitive**.

The variables `lastName` and `lastname`, are two different variables:

```
var lastname, lastName;
lastName = "Doe";
lastname = "Peterson";
```

Try it Yourself »

JavaScript does not interpret **VAR** or **Var** as the keyword **var**.

JavaScript and Camel Case

Historically, programmers have used different ways of joining multiple words into one variable name:

Hyphens:

first-name, last-name, master-card, inter-city.

Hyphens are not allowed in JavaScript. They are reserved for subtractions.

Underscore:

first_name, last_name, master_card, inter_city.

Upper Camel Case (Pascal Case):

FirstName, LastName, MasterCard, InterCity.

Lower Camel Case:

JavaScript programmers tend to use camel case that starts with a lowercase letter:

firstName, lastName, masterCard, interCity.

JavaScript Character Set

JavaScript uses the **Unicode** character set.

Unicode covers (almost) all the characters, punctuations, and symbols in the world.

For a closer look, please study our Complete Unicode Reference.

4. Java Script programming Constructs:

. JAVASCRIPT PROGRAMMING CONSTRUCTS

Most of the programming languages have a common set of

programming constructs. JavaScript also provides a complete range of basic programming constructs.

JavaScript Conditional Statements

As in any other programming language conditional statements in JavaScript are used to perform different actions based on different conditions. While writing a code you may want to perform different actions for different decisions. For this JavaScript has following conditional statements:

1. If statement - use this statement if you want to execute some code only if a specified condition is true.

Syntax

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

E.g. Using if statement

E.g.

Note: in case If is written in lowercase letters using uppercase letters (IF) will generate a JavaScript error!

2. if...else statement:- This statement is used if you want to execute some code if the condition is true and any another code if the condition is false.

Syntax:

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

}

E.g. Displaying Greeting using if else construct

3. if...else if....else statement - use this statement if you want to select one of many blocks of code to be executed.

Syntax:

```
if (condition1)
{
//code to be executed if condition1 is true
}
else if (condition2)
{
//code to be executed if condition2 is true
}
else
{
//code to be executed if condition1 and condition2 are not true
}
```

E.g. if...else if....else statement

Good Day

Note: While comparing variables you must always use two equals signs next to each other (==)!. In such case code executed only if the specified condition is true.

5. Switch statement – You can use Switch statement to select one of many blocks of code that is to be executed. With the Switch statement you can select from a number of alternatives.

Syntax:

```
switch(value)
{
Case 1:
//This code executed if first alternative is chosed
break;
Case 2;
//This code executed if second alternative is chosed
break;
default://used to take a default action
}
```

Note: You need to use break to come out of the Switch statement once selected case code is executed

Unit V

1. Java Script Array

JavaScript Arrays

JavaScript arrays are used to store multiple values in a single variable.

Example

```
var cars = ["Saab", "Volvo", "BMW"];
```

Try it Yourself »

What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

```
var array_name = [item1, item2, ...];
```

Example

```
var cars = ["Saab", "Volvo", "BMW"];
```

Try it Yourself »

Spaces and line breaks are not important. A declaration can span multiple lines:

Example

```
var cars = [
  "Saab",
  "Volvo",
  "BMW"
];
```

2. Declaration:

Array Declarations

An "array declaration" names the array and specifies the type of its elements. It can also define the number of elements in the array. A variable with array type is considered a pointer to the type of the array elements.

Syntax

declaration:
  declaration-specifiers init-declarator-list$_{opt}$ **;**

init-declarator-list:
    init-declarator
    init-declarator-list **,** *init-declarator*

init-declarator:
    declarator
    declarator **=** *initializer*

declarator:
    pointer$_{opt}$ *direct-declarator*

direct-declarator: /* A function declarator */
    direct-declarator **[** *constant-expression*$_{opt}$ **]**

Because *constant-expression* is optional, the syntax has two forms:

- The first form defines an array variable. The *constant-expression* argument within the brackets specifies the number of elements in the array. The *constant-expression*, if present, must have integral type, and a value larger than zero. Each element has the type given by *type-specifier*, which can be any type except **`void`**. An array element cannot be a function type.
- The second form declares a variable that has been defined elsewhere. It omits the *constant-expression* argument in brackets, but not the brackets. You can use this form only if you previously have initialized the array, declared it as a parameter, or declared it as a reference to an array explicitly defined elsewhere in the program.

In both forms, *direct-declarator* names the variable and can modify the variable's type. The brackets (**[ ]**) following *direct-declarator* modify the declarator to an array type.

Type qualifiers can appear in the declaration of an object of array type, but the qualifiers apply to the elements rather than the array itself.

You can declare an array of arrays (a "multidimensional" array) by following the array declarator with a list of bracketed constant expressions in this form:

type-specifier declarator **[** *constant-expression* **] [** *constant-expression* **]** ...

Each *constant-expression* in brackets defines the number of elements in a given dimension: two-dimensional arrays have two bracketed expressions, three-dimensional arrays have three, and so on. You can omit the first constant expression if you have initialized the array, declared it as a parameter, or declared it as a reference to an array explicitly defined elsewhere in the program.

You can define arrays of pointers to various types of objects by using complex declarators, as described in Interpreting More Complex Declarators.

Arrays are stored by row. For example, the following array consists of two rows with three columns each:

CCopy
```
char A[2][3];
```

The three columns of the first row are stored first, followed by the three columns of the second row. This means that the last subscript varies most quickly.

To refer to an individual element of an array, use a subscript expression, as described in Postfix Operators.

Examples

These examples illustrate array declarations:

CCopy
```
float matrix[10][15];
```

The two-dimensional array named `matrix` has 150 elements, each having **`float`** type.

CCopy
```
struct {
    float x, y;
} complex[100];
```

This is a declaration of an array of structures. This array has 100 elements; each element is a structure containing two members.

CCopy
```
extern char *name[];
```

This statement declares the type and name of an array of pointers to **`char`**. The actual definition of `name` occurs elsewhere.

3. Function Calling and Function Definition

What is Function in JavaScript?

Functions are very important and useful in any programming language as they make the code reusable A function is a block of code which will be executed only if it is called. If you have a few lines of code that needs to be used several times, you can create a function including the repeating lines of code and then call the function wherever you want.

How to Create a Function in JavaScript

1. Use the keyword **function** followed by the name of the function.
2. After the function name, open and close parentheses.
3. After parenthesis, open and close curly braces.
4. Within curly braces, write your lines of code.

Syntax:

```
function functionname()

{

  lines of code to be executed

}
```

Try this yourself:

This code is editable. Click Run to Execute

```
<html>

<head>

    <title>Functions!!!</title>

    <script type="text/javascript">

    function myFunction()

    {
```

```
    document.write("This is a simple function.<br />");
    }
        myFunction();
    </script>
</head>
<body>
</body>
</html>
```

Function with Arguments

You can create functions with arguments as well. Arguments should be specified within parenthesis

Syntax:

```
function functionname(arg1, arg2)

{

  lines of code to be executed

}
```

Try this yourself:

This code is editable. Click Run to Execute

```
<html>
<head>
    <script type="text/javascript">
        var count = 0;
        function countVowels(name)
        {
            for (var i=0;i<name.length;i++)
            {
            if(name[i] == "a" || name[i] == "e" || name[i] == "i" || name[i] == "o" || name[i] == "u")
            count = count + 1;
            }
```

```
        document.write("Hello " + name + "!!! Your name has " + count + " vowels.");
    }
    var myName = prompt("Please enter your name");
    countVowels(myName);
    </script>
</head>
<body>
</body>
</html>
```

JavaScript Return Value

You can also create JS functions that return values. Inside the function, you need to use the keyword **return** followed by the value to be returned.

Syntax:

```
function functionname(arg1, arg2)

{
```

```
  lines of code to be executed

  return val1;

}
```

Try this yourself:

This code is editable. Click Run to Execute

```
<html>

<head>

    <script type="text/javascript">

        function returnSum(first, second)

    {

        var sum = first + second;

        return sum;

    }

    var firstNo = 78;

    var secondNo = 22;
```

```
    document.write(firstNo + " + " +
secondNo + " = " +
returnSum(firstNo,secondNo));
    </script>
</head>
<body>
</body>
</html>
```

4. Dialog Boxes:- Prompt-Confirm:

JavaScript alert - prompt - confirm Dialog Boxes

Last update on February 26 2020 08:07:09 (UTC/GMT +8 hours)

JavaScript alert - Dialog box

Description

alert() is a simple function to display a message to a dialog box (also called alert box). Here is a simple example to display a text in the alert box.

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset=utf-8>
<title>Javascript alert box example-1</title>
</head>
<body>
<h1 style="color: red">JavaScript alert() box example</h1>
<hr />
<script type="text/javascript">
alert("This is a alert box");
</script>
</body>
</html>
```

View the example in the browser

Let's puts some variables in the alert box.

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset=utf-8>
<title>JavaScript alert box example-2</title>
</head>
<body>
<h1 style="color: red">JavaScript alert() box example</h1>
<hr />
<script type="text/javascript">
mess1='Highest marks in Mathematics : ';
math=99;
alert(mess1+math);
</script>
</body>
</html>
```

View the example in the browser

JavaScript prompt - Dialog box

Description

The alert() method can not interact with the visitor. To interact with the user we use prompt(), which asks the visitor to input some information and stores the information in a variable.

See the following web document.

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset=utf-8>
<title>JavaScript prompt() example-2</title>
</head>
<body>
<script type="text/javascript">
visiter_name = prompt("Input your name : ")
if(visiter_name != null && visiter_name != "")
alert("Your Name is : "+visiter_name);
else
alert("Blank name ...!")
</script>
</body>
</html>
```

View the example in the browser

JavaScript confirm - Dialog box

Description

Confirm() displays a dialog box with two buttons, OK and Cancel and a text as a parameter. If the user clicks on OK button, confirm() returns true and on clicking Cancel button, confirm() returns false.

See the following web document.

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset=utf-8>
<title>JavaScript confirm box example </title>
</head>
<body>
<h1 style="color: red">JavaScript confirm() box example</h1>
<hr />
<script type="text/javascript">
mess1='Press Ok to Continue.';
math=99;
x = confirm(mess1);
if (x == true)
{
alert("You have clicked on Ok Button.");
}
else
```

```
{
alert("You have clicked on Cancel Button."); }
</script>
</body>
</html>
```

5. Alert:

JavaScript Popup Boxes

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

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
window.alert("sometext");
```

The `window.alert()` method can be written without the window prefix.

Example

```
alert("I am an alert box!");
```

Try it Yourself »

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.

Syntax

```
window.confirm("sometext");
```

The `window.confirm()` method can be written without the window prefix.

Example

```
if (confirm("Press a button!")) {
  txt = "You pressed OK!";
} else {
  txt = "You pressed Cancel!";
}
```

Try it Yourself »

Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
window.prompt("sometext","defaultText");
```

The `window.prompt()` method can be written without the window prefix.

Example

```
var person = prompt("Please enter your name", "Harry Potter");

if (person == null || person == "") {
  txt = "User cancelled the prompt.";
} else {
  txt = "Hello " + person + "! How are you today?";
}
```

Try it Yourself »

Line Breaks

To display line breaks inside a popup box, use a back-slash followed by the character n.

Example

```
alert("Hello\nHow are you?");
```

6. Text-Boxes and Text-areas:

Text Box

A text box is a rectangular area on the screen where you can enter text. It is a common user interface element found in many types of software programs, such as web browsers, email clients, and word processors. When you click in a text box, a flashing cursor is displayed, indicating you can begin typing. If you are using a tablet, tapping inside a text box may automatically display the on-screen keyboard.
There are two different types of text boxes — text fields and text areas. A text field is small box that allows you to enter a single line of text. It is used for entering basic values, such as a name, number, or short phrase. A text area is a larger box that allows you to enter multiple lines of text. Generally, pressing Enter in a text area will enter a newline character, creating a line break. When you press Enter while typing in a text field, either the cursor will jump to the next field or the value will be submitted.

In HTML, a text field is defined by an <input> tag with the type "text." A text area is defined by a <textarea> tag. Text fields are the most common, as they are used for entering queries in search engines and website search boxes. However, many Web forms, such as user registration and website contact pages often contain both types of text boxes. Regardless of what text boxes an online form contains, you can jump from one to the next by pressing the Tab key.

Text Area

The other item that makes our form-building life easier is the <textarea> form element. Text areas let users enter more than one line of text at a time:

```
form action="showInfo.cgi" method="post">
  Your comments are eagerly awaited: <br />
<textarea name="comments"
    rows="5" cols="45">
</textarea>
  <br />
  <input type="submit" value="Send Data" />
</form>
```

As with all other form elements, <textarea> needs a name= attribute. The new attributes for this element are:

- rows= sets the number of lines of input
- cols= sets the width (in characters) of each line

Any text that you put between the opening <textarea> and closing </textarea> tags will be the initial value of your text area. It will be interpreted as pure text, so you shouldn't put any HTML commands in it.

```
<form action="showInfo.cgi" method="post">
  Your comments are eagerly awaited: <br />
<textarea name="comments"
    rows="5" cols="45">
I love your product because
</textarea>
  <br />
  <input type="submit" value="Send Data" />
</form>
```

7.buttons:

The **button** object in HTML is used to represent a <**button**> element. The getElementById() method is used to get the **button** object. Creating **button** object: The **button** object can be created using **JavaScript**. The document. createElement() method is used to create <**button**> element.

8. Radio-Buttons and Checkboxes:

Radio Button Vs Checkbox in HTML

Radio button: It is generally used in HTML forms. HTML forms are required when you need to collect some data from the site visitors. A radio button is used when you want to select only one option out of several available options.

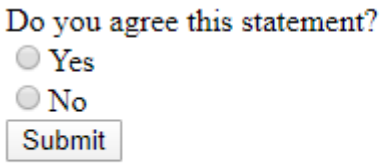
Example:

filter_none
edit
play_arrow
brightness_4

```
<html>

<head>
    <title>
        Radio Button
    <title>
</head>

<body>
    <form>
        Do you agree this statement?
        <br>
        <input type="radio"
               name="agree"
               value="yes">Yes
        <br>
        <input type="radio"
               name="agree"
               value="no">No
        <br>
        <input type="Submit">
    </form>
```

```
</body>

</html>
```

Output:



Checkbox: Checkboxes are also mostly used in HTML forms. A checkbox allows you to choose one or many options to be selected from a list of options.

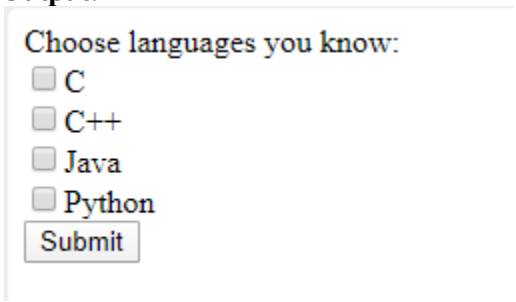
Example:

filter_none
edit
play_arrow
brightness_4

```
<html>

<head>
    <title>
        HTML Checkbox
    <title>
</head>

<body>
    <form>
        Choose languages you know:
        <br>
        <input type="checkbox"
               name="C"
               value="yes">C
        <br>
        <input type="checkbox"
               name="C++"
               value="yes">C++
        <br>
        <input type="checkbox"
               name="Java"
               value="yes">Java
        <br>
        <input type="checkbox"
               name="Python"
               value="yes">Python
        <br>
```

```
        <input type="Submit">
    </form>
</body>

</html>
```

Output:



Difference between radio button and checkbox

RADIO BUTTON	CHECKBOX
It is used when only one option to be selected out of several available options.	Checkbox allows one or many options to be selected.
It is created by using HTML <input> tag but type attribute is set to radio.	It is also created using HTML <input> tag but type attribute is set to checkbox.
It is a single control unit.	It is a multiple control unit.
Radio button is presented as a small circle on the screen.	Checkbox is presented as a small square box on the screen.
Radio button have only 2 states namely- True & False.	Checkbox have 3 states namely- Checked, unchecked & indeterminate.
It is used when you want to limit the users choice to just one option from the range provided.	It is used when you want to allow user to select multiple choices.