How web browsers work

http://www.webdevelopersnotes.com/basics/how_do_web_browser_work.php

HTTP:

HTTP 1.0 document from 1990

http://www.w3.org/Protocols/HTTP/1.0/spec.html

"Definition: HTTP - the Hypertext Transfer Protocol - provides a standard for Web browsers and servers to communicate. The definition of HTTP is a technical specification of a network protocol that software must implement.
HTTP is an application layer network protocol built on top of TCP. HTTP clients (such as Web browsers) and servers communicate via HTTP request and response messages. The three main HTTP message types are GET, POST, and HEAD.
HTTP utilizes TCP port 80 by default, though other ports such as 8080 can alternatively be used……. "
http://compnetworking.about.com/od/networkprotocols/g/bldef_http.htm

HTTPS

The secure hypertext transfer protocol (HTTPS) is a communications protocol designed to transfer encrypted information between computers over the World Wide Web. HTTPS is http using a Secure Socket Layer (SSL). A secure socket layer is an encryption protocol invoked on a Web server that uses HTTPS.

Most implementations of the HTTPS protocol involve online purchasing or the exchange of private information. Accessing a secure server often requires some sort of registration, login, or purchase.

The successful use of the HTTPS protocol requires a secure server to handle the request.

http://msdn.microsoft.com/en-us/library/aa767735(v=vs.85).aspx

HTML

HyperText Markup Language is used for **describing** web documents (web pages) so that they can be correctly **displayed**.

- A markup language is a set of **markup tags**
- HTML documents are described by **HTML tags**
- Each HTML tag **describes** different document content
- http://www.w3schools.com/html/html_intro.asp

Static Web Page

A web page that uses only HTML does not change its appearance or its data and is therefore always a static webpage

URI and URL

In computing, a uniform resource identifier **(**URI**) is a string of characters used to identify a name of a resource. Such identification enables interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols. Schemes specifying a concrete syntax and associated protocols define each URI. The most common form of URI is the uniform resource locator(URL), frequently referred to informally as a *web address.* More rarely seen in usage is the uniform resource name (URN), which was designed to complement URLs by providing a mechanism for the identification of resources in particular namespaces. (Wikipedia)**

URL is an acronym for *Uniform Resource Locator* and is a reference (an address) to a resource on the Internet. It has two main components:

- Protocol identifier: For the URL `http://example.com`, the protocol identifier is `http`.
- Resource name: For the URL `http://example.com`, the resource name is `example.com`.

Note that the protocol identifier and the resource name are separated by a colon and two forward slashes. The protocol identifier indicates the name of the protocol to be used to fetch the resource. The example uses the Hypertext Transfer Protocol (HTTP), which is typically used to serve up hypertext documents. HTTP is just one of many different protocols used to access different types of resources on the net. Other protocols include File Transfer Protocol (FTP), Gopher, File, and News.

The resource name is the complete address to the resource. The format of the resource name depends entirely on the protocol used, but for many protocols, including HTTP, the resource name contains one or more of the following components:

Host Name	The name of the machine on which the resource lives.
Filename	The pathname to the file on the machine.
Port Number	The port number to which to connect (typically optional).
Reference	A reference to a named anchor within a resource that usually identifies a specific location within a file (typically optional).

http://docs.oracle.com/javase/tutorial/networking/urls/definition.html

XML and XSLT

XML can be thought of as an extension of html where the programmer can invent their own tags. XML is about describing the data a website displays. If data is going to be displayed dynamically it can be stored in an external file.

http://www.w3schools.com/xml/xml_whatis.asp

http://www.w3schools.com/xml/xml_usedfor.asp

http://www.w3schools.com/xsl/default.asp

XSLT is a programming language for processing XML data — that is, transforming XML documents. As such, it supports the following:

- A small set of flexible data types: Boolean, number, string, node-set, and external objects.
- A full set of operations: `<xsl:template>`, `<xsl:apply-templates>`, `<xsl:sort>`,`<xsl:output>`, and so on.
- Programming flow-control: `<xsl:if>`, `<xsl:for-each>`, `<xsl:choose>`, and so on.
- http://msdn.microsoft.com/en-us/library/ms767587(v=vs.85).aspx

Javascript

http://www.build-your-website.co.uk/starting-javascript/

Cascading Style Sheet

CSS was first developed in 1997, as a way for Web developers to define the look and feel of their Web pages. It was intended to allow developers to separate content from design so that HTML could perform more of the function that it was originally based on - the markup of content, without worry about the design and layout.

http://webdesign.about.com/od/beginningcss/a/aa021607.htm

Client Side Scripts

Client-side scripting generally refers to the class of computer programs on the web that are executed *client-side*, by the user's web browser, instead of *server-side* (on the web server).[1] This type of computer programming is an important part of the Dynamic HTML (DHTML) concept, enabling web pages to be scripted; that is, to have different and changing content depending on user input, environmental conditions (such as the time of day), or other variables.

http://en.wikipedia.org/wiki/Client-side_scripting

CSS, XML, XSLT and Javascript are used in conjunction with HTML and the web pages can be dynamic to some extent. The scripts are client side scripts

How Domain Name Servers Work

http://computer.howstuffworks.com/dns.htm

TCP/IP

Short for **Transmission Control Protocol/Internet Protocol, TCP/IP** also commonly abbreviated as **TCP** was developed in 1978 and driven by Bob Kahn and Vint Cerf.

Today, TCP/IP is a language governing communications among all computers on the Internet.
TCP/IP is two separate protocols, TCP and IP, that are used together. The Internet Protocol standard dictates how packets of information are sent out over networks. IP has a packet-addressing method that lets any computer on the Internet forward a packet to another computer that is a step (or more) closer to the packet's recipient. The Transmission Control Protocol ensures the reliability of data transmission across Internet connected networks. TCP checks packets for errors and submits requests for re-transmissions if errors are found.

http://www.computerhope.com/jargon/t/tcpip.htm

TCP Characteristics

The following are the ways that I would best describe the Transmission Control Protocol and how it performs the functions described in the preceding topic:

- **Connection-Oriented:** TCP requires that devices first establish a connection with each other before they send data. The connection creates the equivalent of a circuit between the units, and is analogous to a telephone call. A process of negotiation occurs to establish the connection, ensuring that both devices agree on how data is to be exchanged.

- **Bidirectional:** Once a connection is established, TCP devices send data bidirectionally. Both devices on the connection can send and receive, regardless of which of them initiated the connection.

- **Multiply-Connected and Endpoint-Identified:** TCP connections are identified by the pair of sockets used by the two devices in the connection. This allows each device to have multiple connections opened, either to the same IP device or different IP devices, and to handle each connection independently without conflicts.

- **Reliable:** Communication using TCP is said to be *reliable* because TCP keeps track of data that has been sent and received to ensure it all gets to its destination. As we saw in the previous topic, TCP can't really "guarantee" that data will always be received. However, it ***can*** guarantee that all data sent will be checked for reception, and checked for data integrity, and then retransmitted when needed. So, while IP uses "best effort" transmissions, you could say TCP *tries harder*, as the old rent-a-car commercial goes.

- **Acknowledged:** A key to providing reliability is that all transmissions in TCP are acknowledged (at the TCP layer—TCP cannot guarantee that all such transmissions are received by the remote application). The recipient must tell the sender "yes, I got that" for each piece of data transferred. This is in stark contrast to typical messaging protocols where the sender never knows what happened to its transmission. As we will see, this is fundamental to the operation of TCP as a whole.

- **Stream-Oriented:** Most lower-layer protocols are designed so that to use them, higher-layer protocols must send them data in blocks. IP is the best example of this; you send it a message to be formatted and it puts that message into a datagram. UDP is the same. In contrast, TCP allows applications to send it a continuous stream of data for transmission. Applications don't need to worry about making this into chunks for transmission; TCP does it.

- **Data-Unstructured:** An important consequence of TCP's stream orientation is that there are no natural divisions between data elements in the application's data stream. When multiple messages are sent over TCP, applications must provide a way of differentiating one message (data element, record, etc.) from the next.

- **Data-Flow-Managed:** TCP does more than just package data and send it as fast as possible. A TCP connection is *managed* to ensure that data flows evenly and smoothly, with means included to deal with problems that arise along the way.

http://www.tcpipguide.com/free/t_TCPCharacteristicsHowTCPDoesWhatItDoes.htm

Network services will use a network socket **port number** to make a connection and then transfer data. In this way an operating system will be able to pass on the communication to the right program.

File Transfer Protocol (FTP)

What is FTP?

FTP is a set of rules that networked computers to transfer files to and from each other. FTP works on a client-server architecture.

Typically a web hosting site (the SERVER) will have FTP server software. Someone wishing to upload/download files to/from the SERVER would have to have FTP client software installed on their computer (the CLIENT). Internet Explorer has FTP client software in it. The CLIENT would connect to the SERVER site and attempt to open an FTP service on a particular port number (port 21)

Using FTP to manage a website: http://howstuffworks2012.blogspot.co.uk/2012/03/whats-ftp-how-do-i-use-it.html

Transport Characteristics

- Network communications happen in one of two modes. One is "connection-oriented," the other "connectionless." In the connection-oriented model, the two sides establish a session and send messages and data back and forth in the context of the session. In connectionless communication, each transaction is a request from the client followed by a response from the server and nothing more. FTP is connection-oriented. It establishes not one, but two sessions. The first connection carries commands and responses to those commands, while the second connection is a channel for passing files.

Packets

- The file transfer connection does not operate as a stream. That is, the contents of the file are not transferred in a continual unit. As with any Internet application, the data is broken up into smaller segments and transferred in a structure called a packet. Each packet's receipt is acknowledged by the receiver. The header of the packet includes information on the position of the current segment in the overall stream, enabling the receiver to request that a packet get resent in the case of missing data. The receiving FTP program is also able to use this information to re-sequence data that arrives out of order.

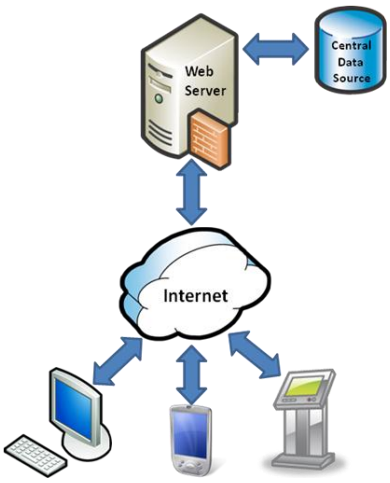
Read more : http://www.ehow.com/info_10073410_describe-characteristics-ftp-protocol.html

Server side scripting

Server-side scripting is a technique used in website design which involves embedding scripts in an HTML source code which results in a user's (client's) request to the server website being handled by a script running on the server-side before the server responds to the client's request. Scripts can be written in any of a number of server-side scripting languages that are available

http://en.wikipedia.org/wiki/Server-side_scripting

……Web pages such as PHP, ASP, and JSP pages are dynamic Web pages. These pages contain "server-side" code, which allows the server to generate unique content each time the page is loaded. For example, the server may display the current time and date on the Web page. It may also output a unique response based on a Web form the user filled out. Many dynamic pages use server-side code to access database information, which enables the page's content to be generated from information stored in the database. Websites that generate Web pages from database information are often called database-driven websites.



http://pc.net/helpcenter/answers/static_and_dynamic_web_pages

Java Servlets are an efficient and powerful solution for creating dynamic content for the Web. Over the past few years Servlets have become the fundamental building block of mainstream server-side Java.

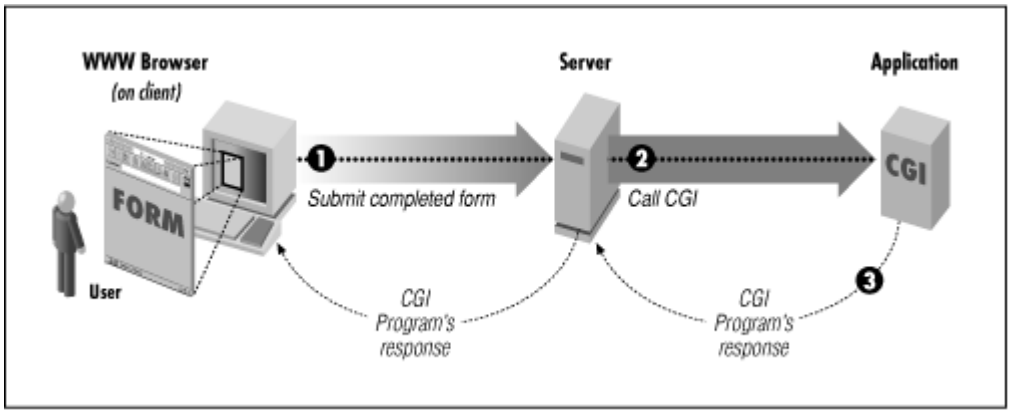
http://www.informit.com/articles/article.aspx?p=170963

Servlets and Java Server Pages are complementary APIs, both providing a means for generating dynamic Web content. A servlet is a Java class implementing the javax.servlet.Servlet interface that runs within a Web or application server's servlet engine, servicing client requests forwarded to it through the server. A Java Server Page is a slightly more complicated beast.

http://www.devx.com/tips/Tip/25217

CGI

CGI is the part of the Web server that can communicate with other programs running on the server. With CGI, the Web server can call up a program, while passing user-specific data to the program (such as what host the user is connecting from, or input the user has supplied using HTML form syntax). The program then processes that data and the server passes the program's response back to the Web browser.



http://www.oreilly.com/openbook/cgi/ch01_01.html