How the Web Functions

A web browser is an application. The application layer provides the protocols that a program needs in order to communicate with another computer. These protocols are destined for a specific port on another computer in the network. For example, the HTTP protocol will arrive on port 80, signifying to the server that the client, the browser, is requesting a web page. When a link is clicked on, a request is sent from the client to a server, specifically termed a web server. In order for the request to be delivered to the correct server, the IP address of the web server must be resolved. The link contains a domain name (techtonicgroup) which is a readable version of the IP address, a nine digit number that uniquely identifies a computer on the internet and represents where to transport the data. The Domain Name System (DNS) is used to find the associated IP address. First the computer will check its local DNS cache, a local storage of information that a computer has previously requested. If the DNS query has not yet expired from the cache, then its IP address is returned. If it did not find the IP address in its local DNS cache due to the DNS query expiring or not having been requested before, a new DNS query is sent to a DNS server. The DNS server, also referred to as a resolver, has a its own local cache, of common requests. If the IP address is found here, it will return it to the client. If it is not found, the DNS server will send a query to a name server. The name server will look at the top level domain, in this case it is .com, which will then send the query to a top level domain (TLD) name server. The TLD name server will forward that query to an authoritative name server which will provide the IP address for a specific domain. This IP address is returned to the DNS server and client and is also stored in the client's local cache and the DNS server's cache. A connection between the client and web server can now be established. Since this follows the HTTPS protocol, a secure connection is first established. HTTPS is an extension of HTTP and the 'S' represents secure. HTTP sends requests in plain text, which means that anyone who can access the communication between two computers can read data transported between two computers. The secure part of the HTTPS involves encrypting data, using an encryption protocol called Secure Socket Layer/ Transport Layer Security (SSL/TLS). A secure connection is established on a specific port on the web server (port 443). A process called public key infrastructure (PKI) is used. This is where both a symmetric cryptography (a single key is used for encryption and decryption) and asymmetric cryptography (a private key and public key are constructed. When the public key is used for encryption, the private key is the only one that can decrypt) are used. The client sends a hello message to the server. This hello message contains information on the version of SSL and its cipher capabilities (information on what encryption algorithms, hashing algorithms it can do). Based on what the client is capable of, the server selects the cipher methods that it is also able to perform. Both the server and client come to an agreement on how data should be encrypted. Next the server will send its Secure Socket Layer (SSL) certificate to the client. This certificate verifies that the the domain name is indeed owned by the owner as well as sharing its public key. The server retains its private key. The client checks the validity of the certificate using Certificate Authorities (CAs). After verification, the client generates a key, called a session key, to be used in a symmetric cryptography. This key is encrypted using the public key from the server. This encrypted session key is sent to the server. The server's private key will decrypt the encrypted session key. The server can now decrypt data sent by the client through symmetric cryptography. HTTP requests can now be sent in a secure fashion from the client. They are encrypted by the session key, sent to the server which will decrypt and then encrypt the response using the same session key back to the client. An HTTP request consists of the URL, methods and other information. With this encryption, this information in the HTTP request is not in plain text format or is not easily readable. Integral information such as what method was requested, parameters, are encrypted. When a web server receives an HTTP request, it will determine if the request requires a dynamic page or a static page. In this case our method will be a GET request (get the home page for https://www.techtonicgroup.com/). Static content is content that does not change or need any kind of modification. Some examples are CSS file, JS files, images, videos. This differs from dynamic content, where a user's input or preferences will result in content needing to be processed and modified before it is return back to the client. If a static page is requested, the web server can send this to the client. If a dynamic page is requested, this request is passed on to the application server. It interacts with a database to generate an html file that is sent back to the web server which then forwards it to the browser. This response can have different statuses, some examples are that it successfully found (200 - OK) the web page or a (400 - not found) if the file was not found.

When the HTML document is returned to the client from the web server, the next step is to parse this document. The browser is an application or piece of software that consists of a number of different components. One of these parts is called a rendering engine, there are different rendering engines (like Webkit for Safari, and Blink for Chrome) based on which browser is used. The responsibility of this engine is to display the webpage. In order to display the web page, the page must be parsed. These rendering engines have a parser generator (such as Bison, Flex). The result of parsing is a tree. Parsing is made up of two components, a parser and a lexer. The lexer will create tokens based on the vocabulary of the language (for example, the h1 tag is a vocab of the HTML language). The parser will check that this vocabulary follows the syntax or grammar rules of the language (for the same example, the h1 tag has an opening and closing tag) and constructs a tree from the tokens as long as the tokens follow the syntax rule else a syntax error is returned. The tree that is constructed from parsing an HTML document is called a document object model (DOM). The nodes of this tree are the tags found in the HTML document, these nodes can have children (other nodes) or lead to values or comments (which would be identified as the leaves). The CSS parser will create a CSSOM (CSS object model). Depending on which browser is used, the combined DOM and CSSOM can be called a render tree or frame tree. The render tree contains nodes that will be displayed or rendered in the browser, so for example, the head node from the DOM tree is not included in the render tree because its content will not be displayed in the browser. JavaScript (JS) is used to manipulate the DOM and CSSOM by accessing the properties of a node. This is what makes a webpage interactive. For example: document.getElementsByTagName(String) will access the

DOM and returns a collection of all the elements with the specified String tag name. This collection can then be modified by using the style property to change the color or other CSS property. Following construction of the render tree, layout is specified, using information in the head of the html document, specified by the meta tag with a name and value of viewport. The “width=device-width” will specify the width of the page on a device. This “width=device-width” sets the context for the CSS formatting (how large an area needs to be and position for content with respect to the width of the device). These areas are then painted, or pixels fill in these areas so that the content can now be viewed in the browser. HTML and CSS are rendered, displayed, in the browser because the nodes that make up the render tree are painted on the screen and these nodes contain HTML content and CSS styles.

Server side code tailors to a user’s experience on the web. Based on a user’s usage of a web page and preferences, a web page becomes customized to the needs of a user. Some examples include based on information collected on the user, a website can modify its responses to the user (for instance Amazon provides suggestions based on previous searches or purchases). Another example is sending alerts that are specific for a user (like notifying that a user has a new message in his or her Facebook messages).

The client side code is executed by the browser. Instead of requesting from the server, it performs an action on its own. It plays an important role in managing the user display, for instance in making a page interactive by using JS, which manipulates the DOM and CSSOM.

Within the HTML document, elements containing the link or script tag, and anything that is external to the HTML document are considered assets. Depending on the number of external links or files, that a web page needs is equivalent to the number of client-side assets.

During installation of server software, one instance of a server is created. Multiple instances can be created but this depends on the needs of a business. For example a business might use multiple instances for performing rolling upgrades. This is where an instance is taken from a cluster of instances, it is upgraded, and then returned to the cluster. This allows the server to continue of completing requests while it upgrades.

Runtime is the when the lines of code within a program are executed. The loader of an operating system will first load a program into memory by going through the machine code of a program. For example, when writing source code in a high level language, like Java, a compiler is required in order to turn that high level language into machine code or binary values so that the computer can run the program. Once in memory, the lines of the program are executed by passing control from the operating system to the code, this is called entry point.

There is at least one instance of the database because during installation a default instance is created. Installation can be performed again to create another instance. The number of instances is dependent on the needs of a business.