How the web functions...

Because the Internet is a global network of computers each computer connected to the Internet **must**have a unique address. Internet addresses are in the form **nnn.nnn.nnn.nnn** where nnn must be a number from 0 - 255. This address is known as an IP address.

If you connect to the Internet through an Internet Service Provider (ISP), you are usually assigned a temporary IP address for the duration of your dial-in session. If you connect to the Internet from a local area network (LAN) your computer might have a permanent IP address or it might obtain a temporary one from a DHCP (Dynamic Host Configuration Protocol) server. In any case, if you are connected to the Internet, your computer has a unique IP address.

So your computer is connected to the Internet and has a unique address. How does it 'talk' to other computers connected to the Internet? An example should serve here: Let's say your IP address is 1.2.3.4 and you want to send a message to the computer 5.6.7.8. The message you want to send is "Hello computer 5.6.7.8!". Obviously, the message must be transmitted over whatever kind of wire connects your computer to the Internet. Therefore the message must be translated from alphabetic text into electronic signals, transmitted over the Internet, then translated back into alphabetic text. How is this accomplished? Through the use of a **protocol stack**. Every computer needs one to communicate on the Internet and it is usually built into the computer's operating system (i.e. Windows, Unix, etc.). The protocol stack used on the Internet is referred to as the TCP/IP.

But what if you don't know the IP address of the computer you want to connect to? What if the you need to access a web server referred to as *www.anothercomputer.com*? How does your web browser know where on the Internet this computer lives? The answer to all these questions is the **Domain Name Service** or **DNS**. The DNS is a distributed database which keeps track of computer's names and their corresponding IP addresses on the Internet.

One of the most commonly used services on the Internet is the World Wide Web (WWW). The application protocol that makes the web work is **Hypertext Transfer Protocol** or **HTTP**. Do not confuse this with the Hypertext Markup Language (HTML). HTML is the language used to write web pages. HTTP is the protocol that web browsers and web servers use to communicate with each other over the Internet. It is an application level protocol because it sits on top of the TCP layer in the protocol stack and is used by specific applications to talk to one another. In this case the applications are web browsers and web servers.

HTTP is a connectionless text based protocol. Clients (web browsers) send requests to web servers for web elements such as web pages and images. After the request is serviced by a server, the connection between client and server across the Internet is disconnected. A new connection must be made for each request. Most protocols are connection oriented. This means that the two computers communicating with each other keep the connection open over the Internet. HTTP does not however. Before an HTTP request can be made by a client, a new connection must be made to the server.

When you hit "https://www.techtonicgroup.com" into a web browser, this is what happens:

1. The browser first connects to a domain name server and retrieves the corresponding IP address for the web server.
2. The web browser connects to the web server and sends an HTTP request (via the protocol stack) for techtonicgroup.com.
3. The web server receives the request and checks for the main web page for techtonicgroup.com. If the page exists, the web server sends it. If the server cannot find the requested page, it will send an HTTP 404 error message. (404 means 'Page Not Found' as anyone who has surfed the web probably knows.)
4. The web browser receives the page back and the connection is closed.
5. The browser then parses through the page and looks for other page elements it needs to complete the web page. These usually include images, applets, etc.
6. For each element needed, the browser makes additional connections and HTTP requests to the server for each element.
7. When the browser has finished loading all images, applets, etc. the page will be completely loaded in the browser window.

**From start to finish how does that data reach you to be rendered in the browser?**

1. You **type an URL** into address bar in your preferred browser.
2. The browser **parses the URL** to find the protocol, host, port, and path.
3. It **forms a HTTP request** (that was most likely the protocol)
4. To reach the host, it first needs to **translate** the human readable host **into an IP number**, and it does this by doing a DNS lookup on the host
5. Then a **socket needs to be opened** from the user’s computer to that IP number, on the port specified (most often port 80)
6. When a connection is open, the **HTTP request is sent** to the host
7. The host **forwards the request** to the server software (most often Apache) configured to listen on the specified port
8. The **server inspects the request** (most often only the path), and **launches the server plugin needed** to handle the request (corresponding to the server language you use, PHP, Java, .NET, Python?)
9. The plugin gets access to the full request, and starts to prepare a HTTP response.
10. To construct the response a **database** is (most likely) **accessed**. A database search is made, based on parameters in the path (or data) of the request
11. Data from the database, together with other information the plugin decides to add, is **combined into a long string** of text (probably HTML).
12. The plugin **combines** that data with some meta data (in the form of HTTP headers), and **sends the HTTP response** back to the browser.
13. The browser receives the response, and **parses the HTML** (which with 95% probability is broken) in the response
14. A **DOM tree is built** out of the broken HTML
15. **New requests are made** to the server for each new resource that is found in the HTML source (typically images, style sheets, and JavaScript files). Go back to step 3 and repeat for each resource.
16. **Stylesheets are parsed**, and the rendering information in each gets attached to the matching node in the DOM tree
17. **Javascript is parsed and executed**, and DOM nodes are moved and style information is updated accordingly
18. The browser **renders the page** on the screen according to the DOM tree and the style information for each node
19. **You** **see** the page on the screen
20. You get annoyed the whole process was too slow.

Server-side rendering is the most common method for displaying information onto the screen. It works by converting HTML files in the server into usable information for the browser.

Whenever you visit a website, your browser makes a request to the server that contains the contents of the website. The request usually only takes a few milliseconds, but that ultimately depends on a multitude of factors:

* Your internet speed
* the location of the server
* how many users are trying to access the site
* and how optimized the website is, to name a few

Once the request is done processing, your browser gets back the fully rendered HTML and displays it on the screen. If you then decide to visit a different page on the website, your browser will once again make another request for the new information. This will occur each and every time you visit a page that your browser does not have a cached version of.

When we talk about client-side rendering, they’re talking about rendering content in the browser using JavaScript. So instead of getting all of the content from the HTML document itself, you are getting a bare-bones HTML document with a JavaScript file that will render the rest of the site using the browser.