Assignment 3

When a user enters the URL in the browser,

the following things happen

URL has three important details :

1. Protocol that will be used for communication over the network (http/ https/ ftp)

2. Domain Name details

3. method on how it will communicate

Question 1:

Below are the tasks that browser performs,

Browser get the fair idea for the 1st and 3rd detail at one go,

and for domain names things are a bit lengthy but any which ways browser performs it really quick

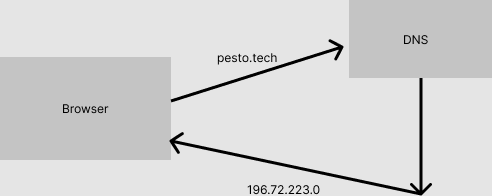
Server on the other end is also a computer, a machine in lamen terms,

it dosent understand alphatets, it understands binary or digit format, called ip addresses

but for humans, things will be very difficult if we have to remember ip addresses for a number of websites, because of which there is a domain name system,

which acts like a directory and has ip addresses for the respective domain name,

\*sample diagram\*



Now here's how the actual process works,

After hitting the URL,

firstly dns query is run on the browser cache, as browser temporarily stores these details for the website that u have visited recently

if no success in the broswer cache, we turn towards router cache

if no success in the above 2 steps then broswer takes help of your resolver server (ISP), DNS query runs in ISP cache

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*not done by the browser\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

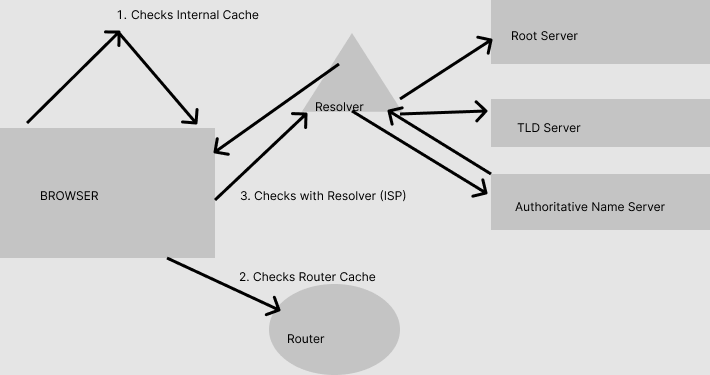
if still no success, then resolver sends query to the root server of the DNS heirarchy, which dosent give us the ip details but

guides us as to where actually will you get the details just like the government employees :)

So for example if you are searching for a top domain like (.com, .org), it will tell you to go to TLD server

TLD server stores the address info and advises that dont worry u will definitely get in authoritative name server

finally authoritative name server provides with the details and resolver stores in it cache and sends it back to the browser



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now Browser initiates TCP connection with the Server

To communicate with server over the network TCP/IP is mostly followed,

A connect'is built using the TCP 3 way handshake mechanism

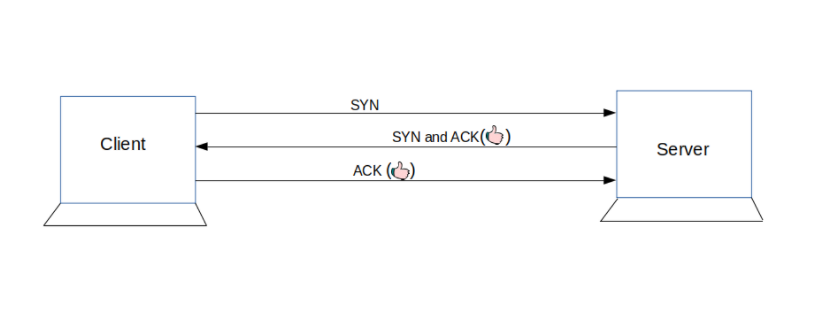
As the name suggests, it has 3 steps,

1. Client sends SYN to check if the other computer is open for connection or not

2. If the other computer is open for connection it sends ACK with a SYN message

3. Now the first computer after receiving message from the other sends the acknowledgment ACK and the connection is established.

\*diagram\*



After the connection is established,

Browser requests for desired info from the server

Question 2 :

Browsers high level components

\*User Interface

\*Browser Engine

\*Rendering Engine

\*Networking

\*UI BackEnd

\*Javascript Interpreter

\*Data Storage

Question 3 :

Rendering Engine and its use

\*Rendering Engine is responsible for displaying a specific web page requested by the user on their screen.

\*The engine draws structured text from a document (often HTML),

and formats it properly based on the given style declarations (often given in CSS).

\*Every browser has its own unique rendering engine.

Google Chrome and Opera v.15+: Blink

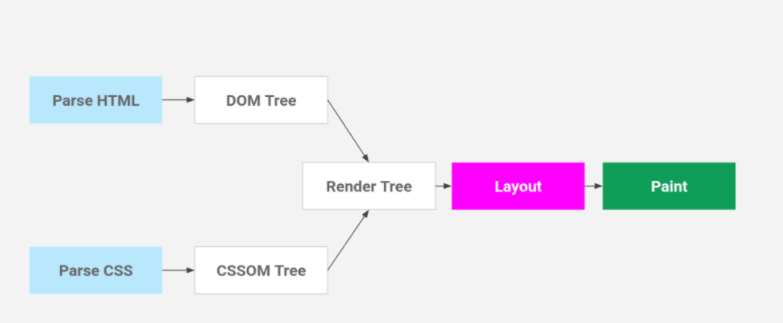
Internet Explorer: Trident

Mozilla Firefox: Gecko

Chrome for iOS and Safari: WebKit

Question 4:

Parsers (HTML, CSS etc)



\*Parsing and Rendering turn the HTML content into a web page with colors and backgrounds and pictures.

HTML Parsing: HTML Text -> Tokenization -> DOM Tree

CSS Parsing: CSS Text -> Tokenization -> CSSOM Tree

DOM and CSSOM are merged to form a Render Tree

\*We type a URL and press the enter and the server responds with index.html.

However, an HTML content is not what we see when we visit a website... we see a web page with colors and backgrounds and animations and pictures.

So there's a process that turns the HTML content to a pretty webpage, and that is parsing and rendering!

Question 5 :

Script Processors

\*Parsing halts when it comes across <script>

\*So if I have <script src="path/to/script"></script> in the middle of the HTML,

The parser will halt there, will fetch the script, wait for the response, execute it and then it will continue the parsing.

This is why we put <scipt> at the end of the body so that we can complete the parsing first.

Question 6 :

Tree Construction

\*For rendering, a DOM and CSSOM are merged to form something called a Render Tree.

Render Tree has the information required to mark and paint elements on the screen.

Also while forming a Render Tree, elements like <head>, <link>, <script>, and elements with 'display: none' in CSS are ignored since they are not rendered on the screen.

\*Note that the elements with 'opacity:0' or 'visibility: none' are included in the render tree,

even though they are not painted on the screen they do take their positions and render as an empty space and thus are required for calculations.

\*So now we have a render tree with all the information that is needed to create a visual page.

Now, the renderer will use this information to create a Layout and then a Paint

Question 7 :

Layout and Painting

\*The layout is where the elements are marked on the screen.

The layout includes all the calculations and mathematics behind an element's position

so it takes all the properties related to the position (height, width, position, top left right bottom, etc)

from The Render Tree and places the elements on the screen.

\*After Layout, a Paint happens. Paint takes properties like color, background-color, border-color, box-shadow, etc. to paint the screen with colors.

\*After the paint, we see the content on the screen and the first time we see something other than a white screen is called 'First Paint'.

The term First Paint is used in performance reports to show how long your website took to show something on the screen.