★ Difference b/w HTTP1.1 VS HTTP2 HTTP version history

i) HTTP 1.1 used to process text commands to complete request-response cycles whereas HTTP 2 use binary protocol rather than text which eliminate security concerns associated with textual nature of HTTP1.1.

ii) HTTP1.1 is order & blocked constrained as only defined no of requests be made at a time & in suitable order whereas HTTP2 is multiplexed protocol which handles parallel requests over same connection

iii) HTTP 1.1 don't support compression of headers whereas HTTP 2 Compresses headers & removes duplication & overhead of data.

iv) HTTP1.1 lacks Server Push whereas HTTP 2 support Server push which allows server to send additional catcheable information to client that isn't requested but is anticipated in future requests.

★ Differences b/w Browser JS VS Node JS.

i) Browser.js is used for frontend while Node.js is used for Backend applications

ii) Node.js has full system access i.e. it can read & write directly to the file system like any other application while Browser.js is sandboxed for safety purposes & have access limited to browser.

iii) In Node.js many objects are missing like —

a) 'window' object

b) 'location' object

c) 'document' object

while browser.js has all these as predefined objects but Browser.js missing on these —

a) "global" object contain several functions that are not available in browser as they are needed for server side work.

b) "require" object which is used to include modules in app

iv) Browser.js runs on any engine like Spidermonkey (firefox), V8 (Google Chrome), Nitro (Safari), Chakra (IE) while Node.js runs in V8 engine used by Google Chrome.

v) Node.js is headless i.e. without any GUI while Browsers are not headless.

★ What happens when you type a URL in address bar in browser?

1) URL is typed in the address bar of browser.

2) The browser checks the cache for a DNS record to find the corresponding IP address of URL.

To check DNS record, browser checks 4 caches:

a) first, it checks browser cache.

b) Second, browser check the OS cache.

c) Third, it checks for Router cache.

d) Fourth, it check for ISP cache.

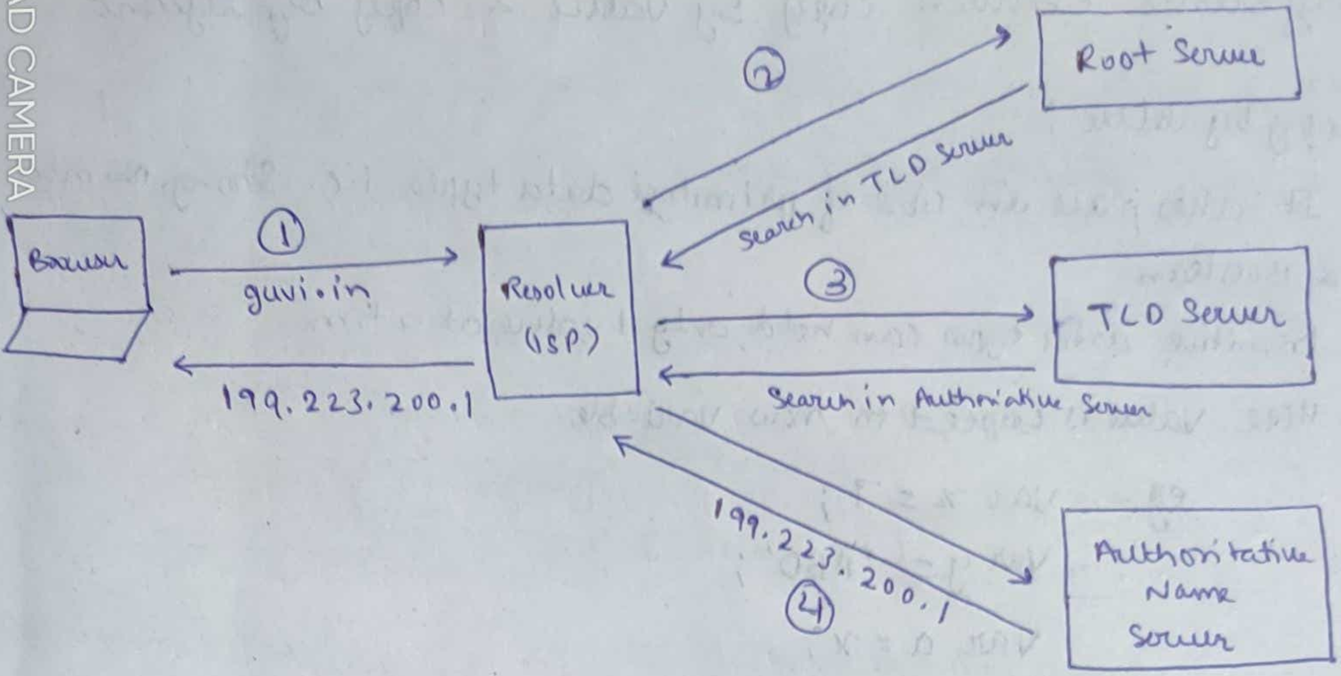
3) If Requested URL is not in cache, ISP's DNS Server initiates a DNS query to find IP address of server that host the url.

i.e. The request is sent to top or Root server of DNS hierarchy. If we are searching IP address of top level domain (.com, .net, .Gov) it tells the resolver server to search TLD Server (Top level Domain).

→ Resolver asks TLD server to give IP address of our domain name. TLD server tells resolver to ask it to Authoritative Name Server.

→ The authoritative name server is responsible for knowing everything about domain name.

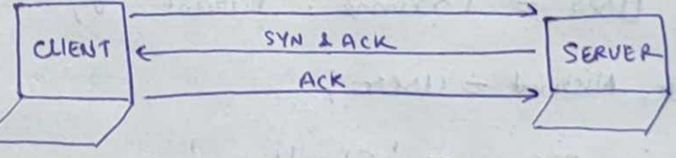
→ Finally resolver (ISP) gets IP address & send it back to browser, & also stores it in cache so that next time, if same query comes then it does not have to go to all step again.



4) TCP connection initiates with the server by Browser

Once IP address is found, connection is initiated.
To communicate over the network, internet protocol is followed.
TCP/IP is most common protocol. A connection is build between two using a process called "TCP 3-way handshake".

→ A client computer sends a SYN message
→ Another computer sends ACK message with SYN message
→ First computer recieves message & acknowledge by sending ACK.



5) The browser sends an HTTP request to the webserver.

6) The server handles the request & sends back response

7) The server sends out an HTTP response.

8) The Browser displays the HTML content (for HTML responses)

* Difference between copy by value & copy by reference

Copy by value :

→ It takes place in case of primitive data types i.e. String, Number & Boolean

→ Primitive data types can hold only 1 value at a time

→ Here value is copied to new variable.

```
eg:-  var x = 7;
      var y = 'ABD';
      var a = x;
      var b = y;
```

here value of x i.e. 7 is copied to a & 'ABD' to b

→ Variables are independent of each other.

Copy by Reference:

→ It takes place is case of composite or non primitive data types i.e. Array, Objects, function

→ Compositive data types can hold collections of values & more complex entities.

→ Here inspite of value address of memory location is passed to new variable.

```
eg:   var user = {name: 'ParasF'};
      var Nishant = user;
      admin.name = 'Choudhary';
```

→ Both user & Nishant store address of memory location

* How to copy by value Composite data type (array + objects)

There are 3 ways to copy by value for composite data types:

1) Using spread (...) operator
2) Using Object.assign() method
3) Using JSON.stringify() and JSON.parse() methods

1) Using Spread :

It spread the elements of that particular array or object & its values can be used to assign to some other variable

eg: let a = [10, 20, 30]

let b = [..a]

a[0] = 99

Console.log(a) // print [99,20,30]

Console.log(b) // print [10,20,30]

eg: In Objects:

let obj1 = { foo: 'bar', x: 42 };

let obj2 = { foo: 'baz', y: 13 };

let clonedobj = {...obj1 }

2) Using Object.assign() :

The Object.assign() method copies all enumerable own properties from one or more source objects to target object.

eg: var a = [1, 2, 3]

var b = Object.assign ([], a)

Console.log(a,b) // [1,2,3] [1,2,3]

b[2]=100

Console.log (a,b) // [1,2,3] [1,2,100]

In Object:

Const target = {a:1, b:2 };

Const source = { b:4, c=5 };

Const ret = Object.assign(target, source);

Console.Log(target); // Object {a:1, b:4, c;5}

Console.Log(ret); // Object {a:1, b:4, c:5}

3) Using JSON.parse() & JSON.stringify()

JSON.parse() takes JSON string & transform it into Javascript object.
JSON.stringify() take Javascript Object & transform it into JSON string.

```
eg:  a = [1,2,3]
     var b = JSON.parse (JSON.stringify (a))
     Console.log (a,b) // [1,2,3] [1,2,3]
      b[2] = 100
     Console.log (a,b) // [1,2,3] [1,2,100]
```