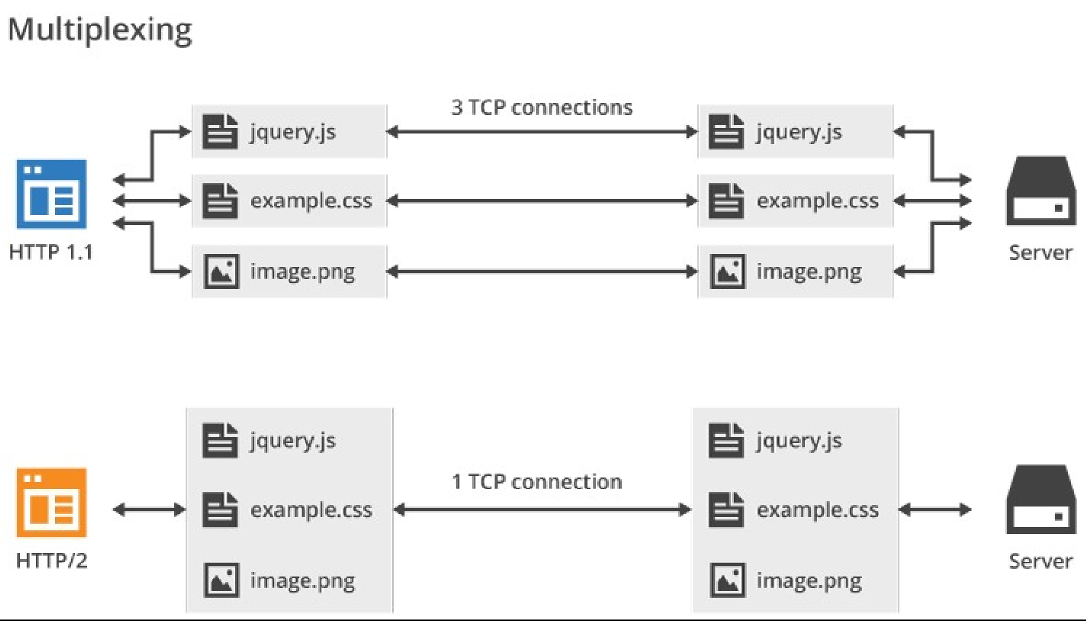
TASK 1- 13th March

1. Difference between HTTP1. 1 vs HTTP2

- The Hypertext Transfer Protocol, or HTTP, is an application protocol that has been the de facto standard for communication on the World Wide Web since its invention in 1989.

SL NO	HTTP1.1	HTTP2
1.	HTTP is a top-level application protocol that exchanges information between a client computer and a local or remote web server	HTTP/2 began as the SPDY protocol, developed primarily at Google with the intention of reducing web page load latency by using techniques such as compression, multiplexing, and prioritization.
2	keeps all requests and responses in plain text format	uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics, such as verbs, methods, and headers.
3	Allows to have **persistent connections** which means that we can have more than one request/response on the same HTTP connection.	Although the design of HTTP/2 effectively addresses the HTTP-transaction-level head-of-line blocking problem by allowing multiple concurrent HTTP transactions, all those transactions are multiplexed over a **single TCP connection,**
4.	Pipelining has been added, allowing to send a second request before the answer for the first one is fully transmitted, lowering the latency of the communication	Uses header compression to reduce overhead, allows servers to "push" responses proactively into client cache



2. http version history

- The Hypertext Transfer Protocol (HTTP) is one of the most ubiquitous and widely adopted application protocols on the Internet: it is the common language between clients and servers, enabling the modern web. From its simple beginnings as a single keyword and document path, it has become the protocol of choice not just for browsers, but for virtually every Internet-connected software and hardware application**.**
- In 1989, while he was working at CERN, Tim Berners-Lee wrote a proposal to build a hypertext system over the Internet. Initially calling it the *Mesh*, it was later renamed to *World Wide Web* during its implementation in 1990. Built over the existing TCP and IP protocols, it consisted of 4 building blocks:
  - ➢ A textual format to represent hypertext documents, the *HyperText Markup Language* (HTML).
  - ➢ A simple protocol to exchange these documents, the *HypertText Transfer Protocol* (HTTP).
  - ➢ A client to display (and accidentally edit) these documents, the first Web browser called *WorldWideWeb*.
  - ➢ A server to give access to the document, an early version of *httpd*.
- **Evolution of HTTP**
  1. **HTTP/0.9 – The one-line protocol**
     The initial version of HTTP had no version number; it has been later called 0.9 to differentiate it from the later versions. HTTP/0.9 is extremely simple: requests consist of a single line and start with the only possible method GET followed by the path to the resource (not the URL as both the protocol, server, and port are unnecessary once connected to the server).
  2. **HTTP/1.0 – Building extensibility**
     - ➢ HTTP/0.9 was very limited and both browsers and servers quickly extended it to be more versatile:
     - ➢ Versioning information is now sent within each request (HTTP/1.0 is appended to the GET line)
     - ➢ A status code line is also sent at the beginning of the response, allowing the browser itself to understand the success or failure of the request and to adapt its behavior in consequence (like in updating or using its local cache in a specific way)
     - ➢ The notion of HTTP headers has been introduced, both for the requests and the responses, allowing metadata to be transmitted and making the protocol extremely flexible and extensible.

     With the help of the new HTTP headers, the ability to transmit other documents than plain HTML files has been added (thanks to the Content-Type header).

3. HTTP/1.1 – The standardized protocol

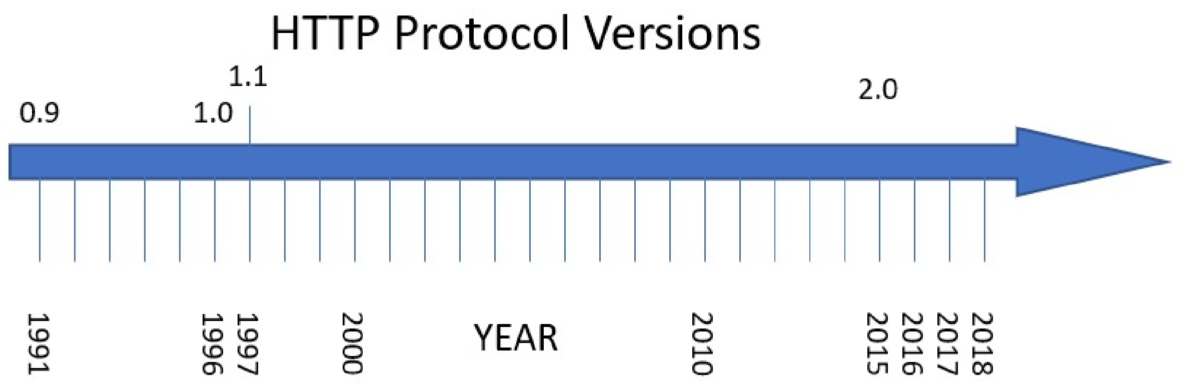
HTTP/1.1 clarified ambiguities and introduced numerous improvements:

- A connection can be reused, saving the time to reopen it numerous times to display the resources embedded into the single original document retrieved.
- Pipelining has been added, allowing to send a second request before the answer for the first one is fully transmitted, lowering the latency of the communication.
- Chunked responses are now also supported.
- Additional cache control mechanisms have been introduced.
- Content negotiation, including language, encoding, or type, has been introduced, and allows a client and a server to agree on the most adequate content to exchange.
- Thanks to the Host header, the ability to host different domains at the same IP address now allows server colocation.

4.HTTP/2 – A protocol for greater performance

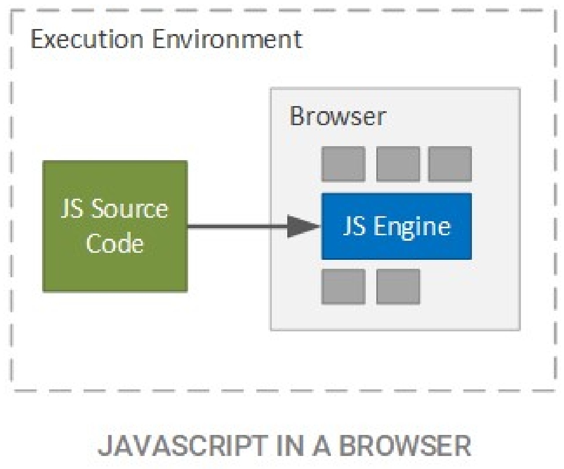
The HTTP/2 protocol has several prime differences from the HTTP/1.1 version:

- It is a binary protocol rather than text. It can no longer be read and created manually. Despite this hurdle, improved optimization techniques can now be implemented.
- It is a multiplexed protocol. Parallel requests can be handled over the same connection, removing the order and blocking constraints of the HTTP/1.x protocol.
- It compresses headers. As these are often similar among a set of requests, this removes duplication and overhead of data transmitted.
- It allows a server to populate data in a client cache, in advance of it being required, through a mechanism called the server push.

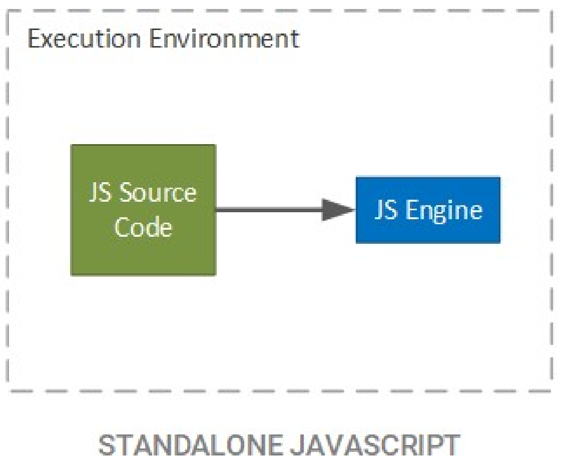


3. List 5 difference between Browser js console vs Node js

It is a javascript runtime based on google chrome javascript engine v8



JAVASCRIPT IN A BROWSER



STANDALONE JAVASCRIPT

SL NO	Browser js	Node js
1	Browser.js is mainly used for client-side applications like validations on a web page or dynamic page display and as the name suggests it gets executed in the browser only	Node.js javascript code gets executed outside the browser as it is an interpreter as well as an environment for running javascript and used for server-side applications
2	Browser.js is used for frontend	Node.js is used for backend applications.
3	Browser.js is sandboxed for the safety purposes and have access limited to the browser.	Node.js has full system access i.e can read and write directly to the file system like any other application that also concludes that we can write complete software using Node.js
4	Browser.js has window, location, document, objects.	Global object(needed for server side), require object (used to include modules in the app.
5	Browser.js runs any engine like Spider monkey (Firefox), JavaScript Core (Safari), V8 (Google Chrome) accordingly to the browser	Node.js runs in a V8 engine which is mainly used by google chrome.
6	Not headless	Node.js is headless i.e without any GUI
7	Moduling is not mandatory for browser javascript.	In Node.js everything is a module i.e it is mandatory to keep everything inside a module

4. What happens when you type a URL in the address bar in the browser?

Steps:

1. You enter the URL in the browser.

Suppose we want to visit the website of **Google**. So we type https://www.google.com/ in the address bar of our browser. When we type any URL we basically want to reach the server where the website is hosted.

2. The browser looks for the IP address of the domain name in the DNS(Domain Name Server).

DNS is a list of URLs and their corresponding IP address just like the telephone book has phone numbers corresponding to the names of the people. We can access the website directly by typing the IP address but imagine remembering a group of numbers to visit any website. So, we only remember the name of the website and the mapping of the name with the IP address is done by the DNS.

The DNS checks at the following places for the IP address.

1. **Check Browser Cache**: The browser maintains a cache of the DNS records for some fixed amount of time. It is the first place to run a DNS query.
2. **Check OS Cache**: If the browser doesn't contain the cache then it requests to the underlying Operating System as the OS also maintains a cache of the DNS records.
3. **Router Cache**: If your computer doesn't have the cache, then it searches the routers as routers also have the cache of the DNS records.
4. **ISP(Internet Service Provider) Cache**: If the IP address is not found at the above three places then it is searched at the cache that ISP maintains of the DNS records. If not found here also, then ISP's **DNS recursive search** is done. In "*DNS recursive search*", a DNS server initiates a DNS query that communicates with several other DNS servers to find the IP address.

So, the domain name which you entered got converted into a DNS number. Suppose the above-entered domain name https://www.google.com/ has an IP address 216.58.193.78 So, if we type 216.58.193.78 in the browser we can reach the website.

3. The Browser initiates a TCP connection with the server.

When the browser receives the IP address, it will build a connection between the browser and the server using the internet protocol. The most common protocol used is TCP protocol. The connection is established using a three-way handshake. It is a three-step process.

1. **Step 1 (SYN):** As the client wants to establish a connection so it sends an SYN(Synchronize Sequence Number) to the server which informs the server that the client wants to start a communication.

2. **Step 2 (SYN + ACK):** If the server is ready to accept connections and has open ports then it acknowledges the packet sent by the server with the SYN-ACK packet.
3. **Step 3 (ACK):** In the last step, the client acknowledges the response of the server by sending an ACK packet. Hence, a reliable connection is established and data transmission can start now.

4.**The browser sends an HTTP request to the server.**

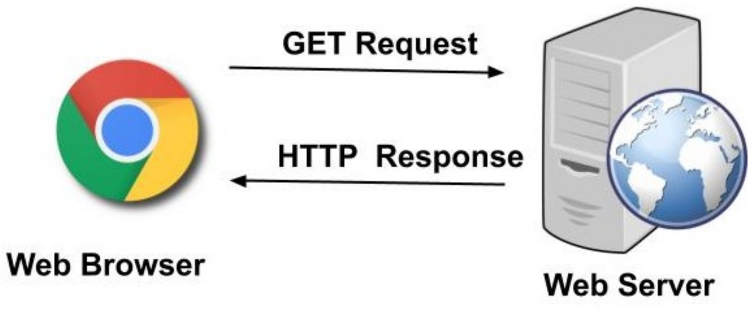
The browser sends a GET request to the server asking for https://www.google.com/ webpage. It will also send the cookies that the browser has for this domain. Cookies are designed for websites to remember stateful information (items in the shopping cart or wishlist for a website like Amazon) or to record the user's browsing history etc. It also has additional information like request header fields(User-Agent) for that allows the client to pass information about the request, and about the client itself, to the server. Other header fields like the Accept-Language header tells the server which language the client is able to understand. All these header fields are added together to form an HTTP request.

5. The server handles the incoming request and sends an HTTP response.

The server handles the HTTP request and sends a response. The first line is called the status line. A Status-Line consists of the protocol version(e.g HTTP/1.1) followed by numeric status code(e.g 200)and its associated textual phrase(e.g OK). The status code is important as it contains the status of the response.

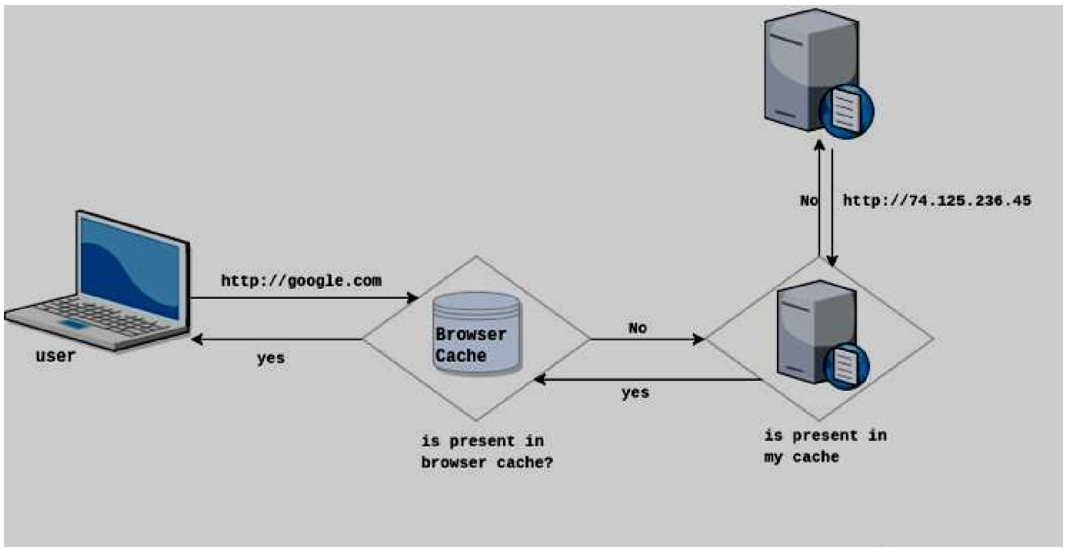
1. 1xx: Informational: It means the request was received and the process is continuing.
2. 2xx: Success: It means the action was successful.
3. 3xx: Redirection: It means further action must be taken in order to complete the request. It may redirect the client to some other URL.
4. 4xx: Client Error: It means some sort of error in the client's part.
5. 5xx: Server Error: It means there is some error on the server-side.

It also contains **response header fields** like Server, Location, etc. These header fields give information about the server. A **Content-Length** header is a number denoting the exact byte **length** of the HTTP body. All these headers along with some additional information are added to form an HTTP response.



6. The browser displays the HTML content.

Now the browser gets the response and the HTML web page is rendered in phases. First, it gets the HTML structure and then it sends multiple GET requests to get the embedded links, images, CSS, javascript files, etc and other stuff. The web page will be rendered and in this case, the https://www.google.com/ web page will be displayed.


No
http://74.125.236.45
http://google.com
user
yes
Browser Cache
No
yes
is present in browser cache?
is present in my cache