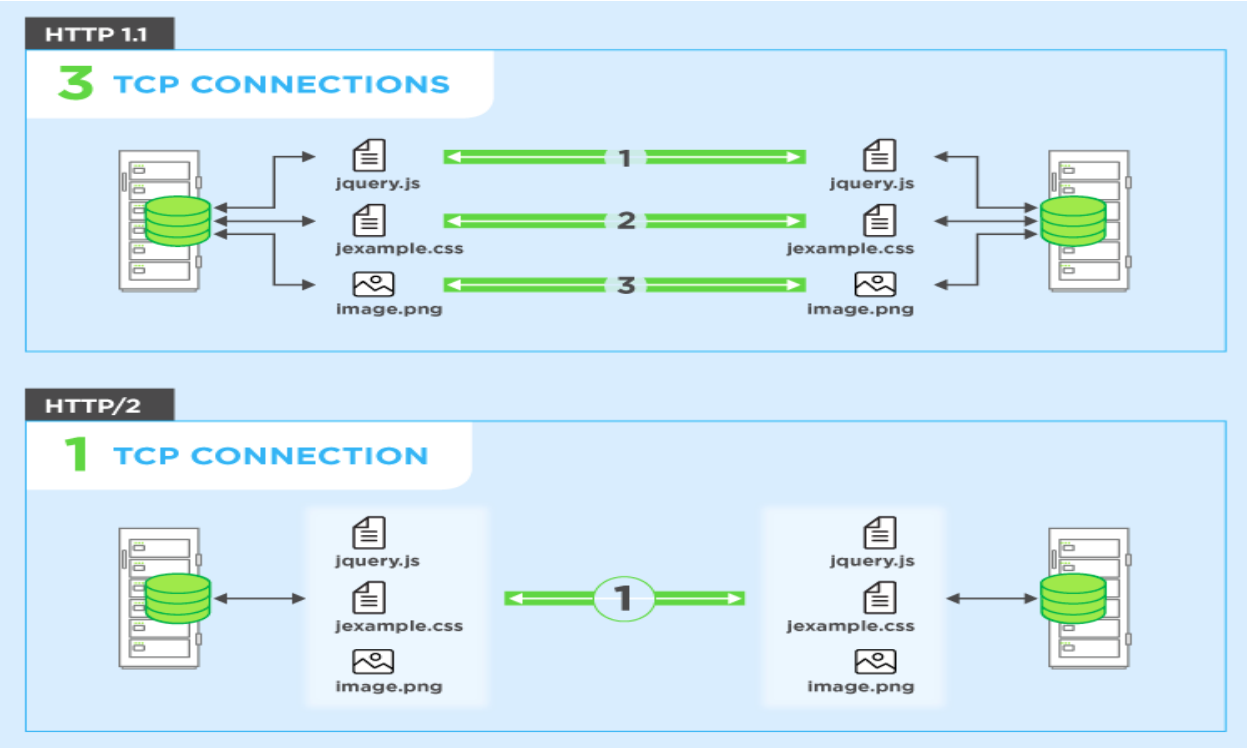
Blog Write-up Task:

1. Difference between HTTP1.1 vs. HTTP2

HTTP is the foundation of data communication in the World Wide Web. It is based on a client server model. It can simply be two computers, 1- Client, the receiver of service, 2 – Server, the provider of service. Given below is the features that sets HTTP2 apart from HTTP1.1.

Request Multiplexing:

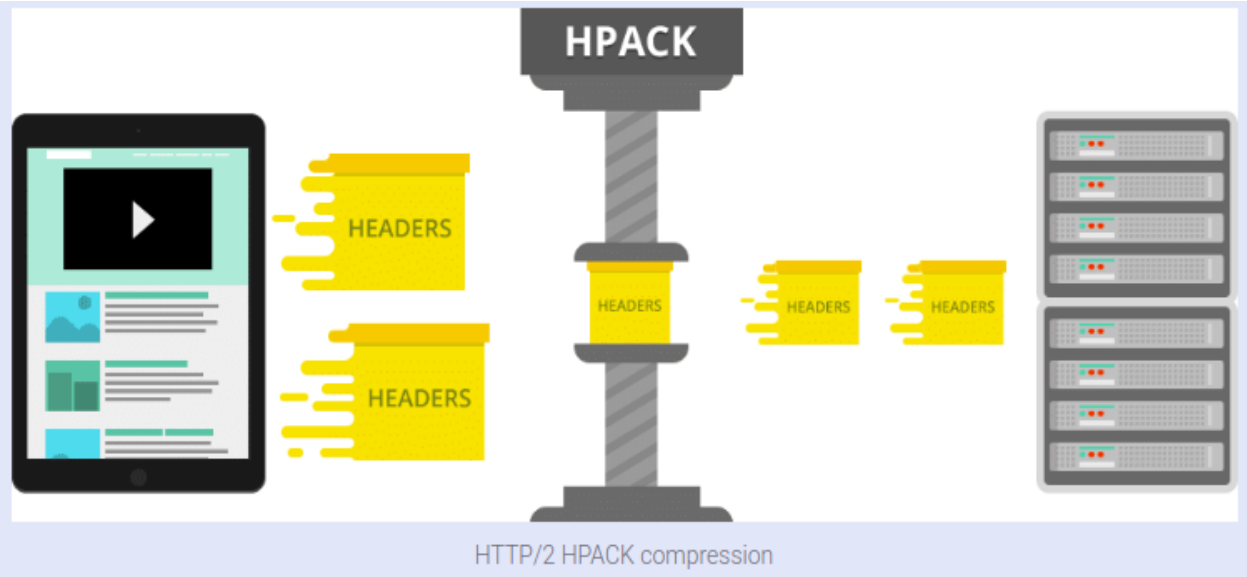
HTTP2 can send multiple requests for data in parallel over a single TCP connection. In HTTP1 we have to send a separate request for each data we need. Request multiplexing reduces the extra time it take to perform the additional round trip making your website load faster.



Header Compression:

HTTP/2 compress a large number of redundant header frames. It uses the HPACK specification as a simple and secure approach to header compression. Both client and server maintain a list of headers used in previous client-server requests.

HPACK compresses the individual value of each header before it is transferred to the server, which then looks up the encoded information in a list of previously transferred header values to reconstruct the full header information.



HTTP/2 HPACK compression

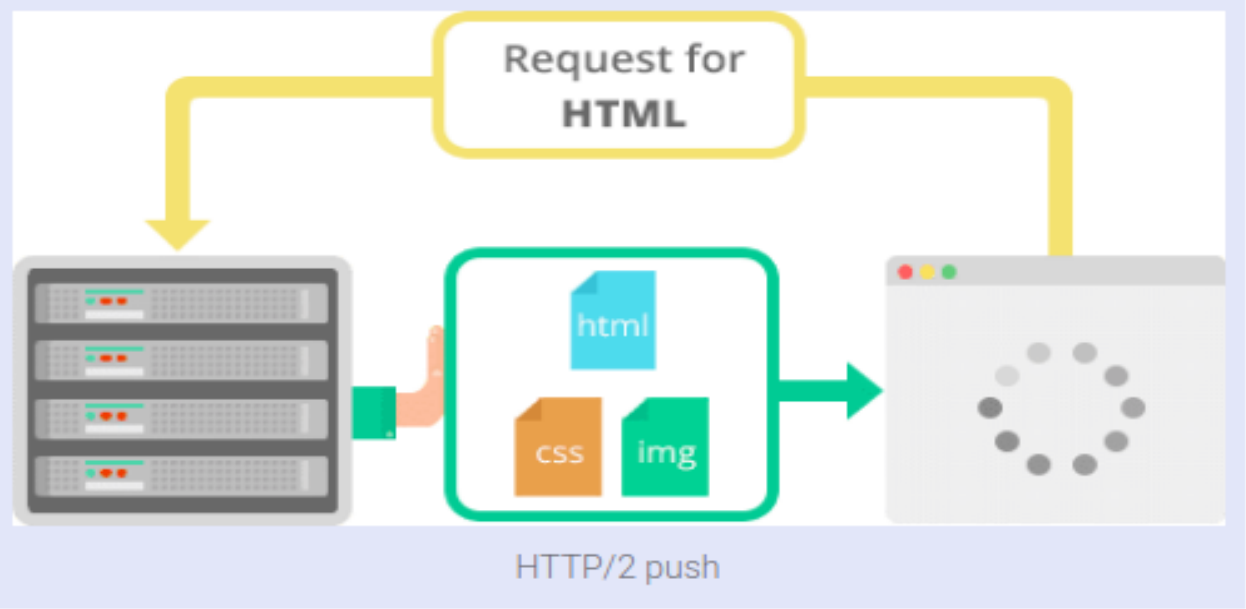
Binary protocol:

The latest HTTP version has evolved significantly in terms of capabilities and attributes such as transforming from a text protocol to a binary protocol. HTTP1.x used to process text commands to complete request-response cycles. HTTP/2 will use binary commands (in 1s and 0s) to execute the same tasks. This attribute eases complications with framing and simplifies implementation of commands that were confusingly intermixed due to commands containing text and optional spaces. Browsers using HTTP/2 implementation will convert the same text commands into binary before transmitting it over the network.

Binary protocols

HTTP2 Server push:

This capability allows the server to send additional cacheable information to the client that isn't requested but is anticipated in future requests. For example, if the client requests for the resource X and it is understood that the resource Y is referenced with the requested file, the server can choose to push Y along with X instead of waiting for an appropriate client request.



2. HTTP version history

HTTP 0.9:

- 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 resource.
- The response is extremely simple and had only the file itself.

HTTP 1.0:

- HTTP version information was sent with each request.
- 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.

- 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.

HTTP 1.1:

- This was the first standardized version of HTTP in 1997.
- 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.

HTTP 2:

- Google demonstrated an alternative way of exchanging data between client and server by implementing SPDY protocol, which increases responsiveness and eliminates duplication of transmitted data. This SDPY was used as a foundation for HTTP2.
- 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.

HTTP 3:

- This version is still work in progress.
- It will use QUIC instead of TCP/TLS in the transport layer.
- QUIC is a low-latency transportation protocol often used for apps and services that require speedy online service. This kind of protocol is a necessity for gamers, streamers or anyone who relies on VoIP in their day-to-day life.

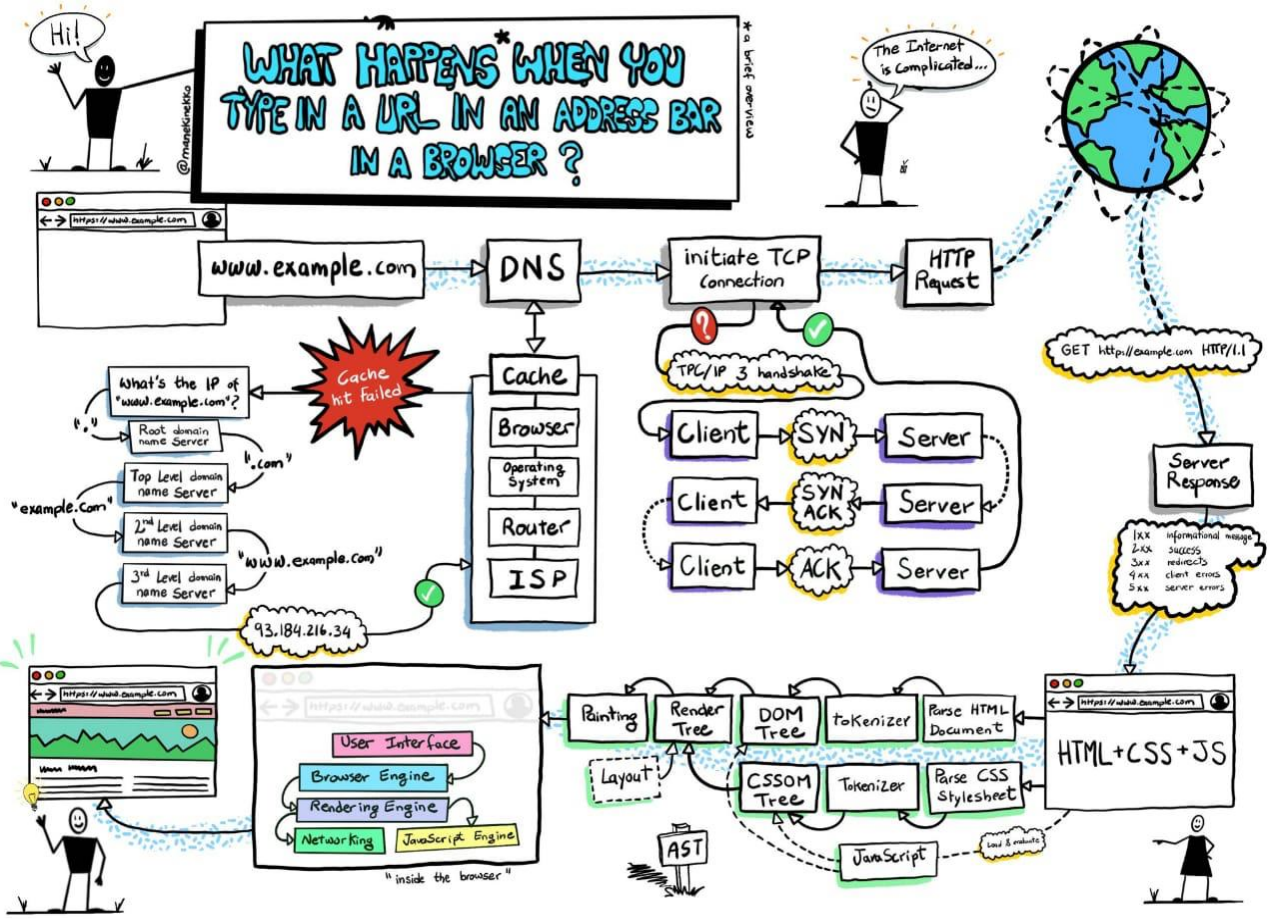
3. List 5 difference between Browser JS console vs. Node JS

Browser JS	Node JS
1. JS is sandboxed in the browser. It cannot execute without the browser.	1. JS is stripped from the browser and can run without the browser.
2. Mainly used in the client side for a web application like attribute validation or providing some dynamic changes in web pages without refreshing the pages.	2. Used in the server side/ backend like creating and executing shell script or accessing any hardware specific information.
3. It runs on any engine like V8, Spider Monkey, JavaScript Core etc.	3. It runs only on V8 engine which mainly uses Google Chrome.
4. Contains objects and elements that interact with the browser like document, object, alert etc.	4. It does not contain objects that interact with the browser. But similar to window object Node JS has global.
5. In browser 'require' is not a pre-defined object and we don't to keep the code in a module.	5. Node JS has a pre-defined object called 'require' and it has to keep the code in module.

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

- When the user types the URL in the browser, it first goes to the DNS (Domain Name System). In DNS the human readable URL is converted to Machine readable IP address.
- The DNS searches in the Cache → Browser → Operating System → Router →ISP and returns the IP address.
- After this step the TCP connection is initiated. TCP is a handshake protocol and so it waits for an acknowledgement for successful connection.
- Once the Connection is done, the HTTP request is sent World Wide Web, with the information sent along with the HTTP request. It then finds the server that will give the required information.

- Once this is done, the server acknowledges with Server codes given below, that help in identify the status of your request.
  - 1xx – Hold on/ Server is busy
  - 2xx – Success
  - 3xx – Unauthorized
  - 4xx – Client side error
  - 5xx – Server side error



- After this is done and if the server code is successful, the server replies with the HTML, CSS and JS information.
- Now the given HTML, CSS is converted to the DOM and CSSOM tree structure by the tokenizer and then it starts rendering the contents of the document.
- Finally the after all these steps, the required document is painted on the browser screen.