Question 1 – List five differences between Browser Js and node Js.

	Browser Js	**Node Js**
1)	Js engine is not a stand-alone system. It functions along with Networking module, Browser engine, rendering engine and other inbuilt systems.	Here, the Js engine is a stand-alone system.
2)	Js engine has a rendering engine inbuilt in the browser along with it.	Js engine has no rendering engine.
3)	Js engine is present along with an inbuilt networking layer in the browser.	Js engine has no networking layer. In order to issue http calls, we have to use third party http packages. We have to source http packages from outside the node Js environment to issue http calls.
4)	Script tag is used to enclose javascript codes. To execute the javascript file, we have to use the browser.	In node js environment, we just use the command 'node' along with file name.js in the node js terminal to execute the .js file.
5)	Output is received in the browser where js files are executed.	We get the ouput in node js terminal itself.

Question 2 – What happens when you type an URL in the address bar of the browser?

The processes which happen after typing an URL in the address bar of the browser are as follows,

Forward Lookup: The URL name is converted to an IP address. This is done with DNS and the process is called Forward lookup. If an URL called http://www.guvi.io is typed, it'll be converted to an IP address like http://234.23.45.6.80/index.html.

The number ‘80’ in the above IP address refers to the predefined port number for the protocol ‘HTTP’. Also, by default, any http request would look for index.html file, as present in the above IP address, in the server and index.html will be the first file to be loaded in the web browser.

http request: The networking module in the browser enables the generation of a http call using GET method. The http calls indicates the request for a specific resource from the server to the client. So the http request reaches the port 80 in the server and searches for the index.html file in the port 80. Once it is found, the response is sent to browser.

The data transaction between server and client happens with TDP or UDP based on what the web application was built.

Rendering engine: Files from the server reach the **browser engine**. From there, the HTML and CSS files are sent to rendering engine and the Js files are sent to **Js engine**. Rendering engine has HTML and CSS parsers which converts the HTML and CSS files into a layout tree. Js engine converts the Js files into machine codes using interpreter, compiler and profiler.

UI Backend: The layout tree undergoes painting by OS specific methods. By painting method, the layout tree is converted to user understandable web page in the browser window.

The above seen processes happen between typing an URL and the eventual web page display in browser window.

Question 3 – HTTP version history

The first HTTP version was HTTP/0.9.

HTTP/0.9 — The One-line Protocol

- Initial version of HTTP — a simple client-server, request-response, telenet-friendly protocol
- Request nature: single-line (method + path for requested document)
- Methods supported: GET only
- Response type: hypertext only
- Connection nature: terminated immediately after the response

- No HTTP headers (cannot transfer other content type files), No status/error codes, No URLs, No versioning

HTTP/1.0 — Building extensibility

- Browser-friendly protocol
- Provided header fields including rich metadata about both request and response (HTTP version number, status code, content type)
- Response: not limited to hypertext (Content-type header provided ability to transmit files other than plain HTML files — e.g. scripts, stylesheets, media)
- Methods supported: GET, HEAD, POST.
- Connection nature: terminated immediately after the response.

Establishing a new connection for each request — major problem in both HTTP/0.9 and HTTP/1.0

Both HTTP/0.9 and HTTP/1.0 required to open up a new connection for each request (and close it immediately after the response was sent). Each time a new connection establishes, a TCP three-way handshake should also occur. For better performance, it was crucial to reduce these round-trips between client and server. HTTP/1.1 solved this with persistent connections.

HTTP/1.1 — The standardized protocol

- This is the HTTP version currently in common use.
- Introduced critical performance optimizations and feature enhancements — persistent and pipelined connections, chunked transfers, compression/decompression, content negotiations, virtual hosting (a server with a single IP Address hosting multiple domains), faster response and great bandwidth savings by adding cache support.

Methods supported: GET , HEAD , POST , PUT , DELETE , TRACE , OPTIONS.

- Connection nature: long-lived

Keep-Alive and Upgrade headers

Keep-Alive header

- The Keep-alive header was used prior to HTTP/1.1 and was obsoleted by HTTP/1.1 making persistent connections the default behavior. Keep-alive header can be used to define policies for long-lived communication between hosts (i.e. allows a connection to stay active until an event occurs). This laid foundation for persistence, reusable connections, pipelining, and many more enhanced capabilities in modern web communication protocols.
- Client, server, or any intermediary can provide information for Keep-alive header independently.
- HTTP pipelining, multiple connections, and many more improvements have been implemented, thanks to the Keep-alive header's behavior.

Upgrade header

- With Upgrade header introduced in HTTP/1.1, it is possible to start a connection using a commonly-used protocol, such as HTTP/1.1, then request that the connection switch to an enhanced protocol type like HTTP/2.0 or WebSockets.
- In an upgraded protocol connection, max parameter (maximum request count) is not present. The upgraded protocol can provide new policies for timeout parameter (if not specifically defined, it uses default timeout value in underlying protocol).

HTTPS

- Hyper Text Transfer Protocol Secure (HTTPS) is the secure version of HTTP. It uses SSL/TLS for secure encrypted communications.
- Originally developed by Netscape in mid-1990s, SSL (Secure Socket Layer) is a cryptographic protocol enhancement to HTTP, which defines how client and server should communicate with each other securely. TLS (Transport Layer Security) is the successor of SSL.
- An HTTPS connection can protect the data transfer from the man-in-the-middle attacks and common security threats by providing bidirectional encryption for communications between a client and server.

SSL/TLS Handshake — major problem in HTTPS

- Although HTTPS is secure by its design, the SSL/TLS handshake process consumes a significant time before establishing an HTTPS connection. It normally costs 1–2 seconds and drastically slows down the startup performance of a website.

HTTP/2.0, HTTP/3.0

All above features are being used by major web servers and browsers today. But modern enhancements like HTTP/2.0, HTTP/3.0 are in experimental stages.

The above seen is a brief history on the various HTTP versions.

Question 4 – HTTP version 1.1 and version 2 – difference.

In particular, HTTP/2 is much faster and more efficient than HTTP/1.1. One of the ways in which HTTP/2 is faster is in how it prioritizes content during the loading process is **prioritisation.**

The other differences between HTTP/2 and HTTP/1.1 that impact performance are

Multiplexing: HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. In contrast, HTTP/2 is able to use a single TCP connection to send multiple streams of data at once so that no one resource blocks any other resource. HTTP/2 does this by splitting data into binary-code messages and numbering these messages so that the client knows which stream each binary message belongs to.

Server push: Typically, a server only serves content to a client device if the client asks for it. However, this approach is not always practical for modern webpages, which often involve several dozen separate resources that the client must request. HTTP/2 solves this problem by allowing a server to "push" content to a client before the client asks for it. The server also sends a message letting the client know what pushed content to expect.

Header compression: Small files load more quickly than large ones. To speed up web performance, both HTTP/1.1 and HTTP/2 compress HTTP messages to make them smaller. However, HTTP/2 uses a more advanced compression method called HPACK that eliminates redundant information in HTTP header packets. This eliminates a few bytes from every HTTP packet. Given the volume of HTTP packets involved in loading even a single webpage, those bytes add up quickly, resulting in faster loading.

The above seen are the differences between HTTP/1.1 and HTTP/2.