Task 1 Date: 7-1-2021

* **Difference between http1.1 vs http2?**

Answer-

**HTTP 1.1 –**

HTTP 1.1 is a top-level application protocol that exchanges information between a client computer and a local or remote web server. In this process, a client sends a text-based request to a server by calling a method like Get or Post. In response, the server sends a resource like an HTML page back to the client

HTTP 1.1 assumes that a TCP connection should be kept open unless directly told to close. This allows the client to send multiple requests along the same connection without waiting for a response to each, greatly improving the performance

In HTTP 1.1 flow control relies on the underlying TCP connection. When this connection initiates, both client and server establish their buffer sizes using their system default settings

In HTTP 1.1 if the developer knows in advance which additional resources the client machine will need to render the page, they can use a technique calledresource inlining toinclude the required resource directly within the HTML document that the server sends in response to the initial Get request

**HTTP 2 –**

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.

In HTTP 2 the binary framing layer encodes requests/responses and cuts them up into smaller packets of information, greatly increasing the flexibility of data transfer.

HTTP 2 solves this problem by allowing the client and server to implement their own flow controls, rather than relying on the transport layer. The application layer communicates the available buffer space, allowing the client and server to set the receive window on the level of the multiplexed streams.

HTTP 2 enables multiple concurrent responses to a client’s initial Get request, a server can send a resource to a client along with the requested HTML page, providing the resource before the client asks for it. This process is called server push*.*

* **HTTP version history?**

Answer-

[**HTTP 0.9**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#http0.9_%E2%80%93_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. It is extremely simple; requests consist of a single line and start with the only possible method Get followed by the path to the resource.

[**HTTP 1.0 –**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#http1.0_%E2%80%93_building_extensibility)

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 behaviour in consequence.

[**HTTP 1.1 –**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#http1.1_%E2%80%93_the_standardized_protocol)

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.

[**HTTP 2 –**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#http2_%E2%80%93_a_protocol_for_greater_performance)

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.

[**Post-HTTP 2**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#post-http2_evolution) **–**

Support of [Alt-Svc](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Alt-Svc) allows the dissociation of the identification and the location of a given resource, allowing for a smarter [CDN](https://developer.mozilla.org/en-US/docs/Glossary/CDN) caching mechanism. The introduction of [Client-Hints](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Client-Hints) allows the browser, or client, to proactively communicate information about its requirements, or hardware constraints, to the server.

[**HTTP 3 - HTTP over QUIC**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#http3_-_http_over_quic) **–**

The QUIC transport protocol has several features that are desirable in a transport for HTTP, such as stream multiplexing, per-stream flow control, and low-latency connection establishment. HTTP 3 provides a transport for HTTP semantics using the QUIC transport protocol and an internal framing layer similar to HTTP 2. HTTP 3 frames are used to manage server push, such as PUSH\_PROMISE, MAX\_PUSH\_ID, and CANCEL\_PUSH.

* **List the 5 difference between Browser JS and Node JS?**

Answer-

|  |  |  |
| --- | --- | --- |
|  | **Browser JS** | **Node JS** |
| 1 | Here, we interact with DOM and web Platforms and has the html which is rendered | Here, don’t have the document, window and all the other objects, because it never have to render anything in a page |
| 2 | Browsers are not headless | Node is headless |
| 3 | Here, uses babel to transform code to be ES5/6/7 compatible | In Node, don’t need to do that |
| 4 | Browsers processes response objects | Node processes request object. |
| 5 | Location is another predefined object in browsers, that has all the information about the URL we have loaded. | Location object is related to a particular URL; that means it is for page specific. So, node doesn't require that. |

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

Answer-

1. Browser checks cache for DNS entry to find the corresponding [IP address](https://www.geeksforgeeks.org/introduction-of-classful-ip-addressing/) of website. It looks for following cache. If not found in one, then continues checking to the next until found
   * Browser Cache, Operating Systems Cache, Router Cache, ISP Cache
2. If not found in cache, ISP’s (Internet Service Provider) DNS server initiates a DNS query to find IP address of server that hosts the domain name  
   The requests are sent using small data packets that contain information content of request and IP address it is destined for
3. Browser initiates a [TCP (Transfer Control Protocol)](https://www.geeksforgeeks.org/tcp-and-udp-in-transport-layer/) connection with the server using synchronize (SYN) and acknowledge (ACK) messages
4. Browser sends an [HTTP](https://www.geeksforgeeks.org/http-non-persistent-persistent-connection/) request to the web server. GET or POST request
5. Server on the host computer handles that request and sends back a response. It assembles a response in some format like JSON, [XML](https://www.geeksforgeeks.org/xml-basics/) and HTML
6. Server sends out an HTTP response along with the status of response
7. Browser displays [HTML](https://www.geeksforgeeks.org/html-tutorials/) content