Do a write up for the followings?

1. Difference between HTTP1.1 vs HTTP 2?

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. From the release of HTTP/1.1 in 1997 until recently, there have been few revisions to the protocol. But in 2015, a reimagined version called HTTP/2 came into use, which offered several methods to decrease latency, especially when dealing with mobile platforms and server-intensive graphics and videos. HTTP/2 has since become increasingly popular, with some estimates suggesting that around a third of all websites in the world support it.

**HTTP/1.**

Developed by Timothy Berners-Lee in 1989 as a communication standardfor the World Wide Web, HTTP 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.

You can think of this exchange of requests and responses as a single *application layer* of the internet protocol stack, sitting on top of the *transfer layer* (usually using the Transmission Control Protocol, or TCP) and *networking layers* (using the Internet Protocol, or IP).

**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. This protocol served as a template for HTTP/2 when the Hypertext Transfer Protocol working group httpbis of the [IETF (Internet Engineering Task Force)](https://www.ietf.org/) put the standard together, culminating in the publication of HTTP/2 in May 2015. From the beginning, many browsers supported this standardization effort, including Chrome, Opera, Internet Explorer, and Safari. Due in part to this browser support, there has been a significant adoption rate of the protocol since 2015, with especially high rates among new sites.

From a technical point of view, one of the most significant features that distinguishes HTTP/1.1 and HTTP/2 is the binary framing layer, which can be thought of as a part of the application layer in the internet protocol stack. As opposed to HTTP/1.1, which keeps all requests and responses in plain text format, HTTP/2 uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics, such as verbs, methods, and headers. An application level API would still create messages in the conventional HTTP formats, but the underlying layer would then convert these messages into binary. This ensures that web applications created before HTTP/2 can continue functioning as normal when interacting with the new protocol.

The conversion of messages into binary allows HTTP/2 to try new approaches to data delivery not available in HTTP/1.1, a contrast that is at the root of the practical differences between the two protocols. The next section will take a look at the delivery model of HTTP/1.1, followed by what new models are made possible by HTTP/2.

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/1.1**

Programs like [gzip](https://www.gzip.org/) have long been used to compress the data sent in HTTP messages, especially to decrease the size of CSS and JavaScript files. The header component of a message, however, is always sent as plain text. Although each header is quite small, the burden of this uncompressed data weighs heavier and heavier on the connection as more requests are made, particularly penalizing complicated, API-heavy web applications that require many different resources and thus many different resource requests. Additionally, the use of cookies can sometimes make headers much larger, increasing the need for some kind of compression.In order to solve this bottleneck, HTTP/2 uses HPACK compression to shrink the size of headers.

**HTTP/2**

One of the themes that has come up again and again in HTTP/2 is its ability to use the binary framing layer to exhibit greater control over finer detail. The same is true when it comes to header compression. HTTP/2 can split headers from their data, resulting in a header frame and a data frame. The HTTP/2-specific compression program [HPACK](https://tools.ietf.org/html/draft-ietf-httpbis-header-compression-12) can then compress this header frame. This algorithm can encode the header metadata using Huffman coding, thereby greatly decreasing its size. Additionally, HPACK can keep track of previously conveyed metadata fields and further compress them according to a dynamically altered index shared between the client and the server.

1. HTTP version History?

**History :**

Tim Berners Lee and his team at CERN gets credit for inventing original HTTP and associated technologies.

1. **HTTP version 0.9 –**This was first version of HTTP which was introduced in 1991.
2. **HTTP version 1.0 –**In 1996, RFC 1945 (Request For Comments) was introduced in HTTP version 1.0.
3. **HTTP version 1.1 –**In January 1997, RFC 2068 was introduced in HTTP version 1.1. Improvements and updates to HTTP version 1.1 standard were released under RFC 2616 in June 1999.
4. **HTTP version 2.0 –**The HTTP version 2.0 specification was published as RFC 7540 on May 14, 2015.
5. **HTTP version 3.0 –**HTTP version 3.0 is based on previous RFC draft. It is renamed as HyperText Transfer Protocol QUIC which is a transport layer network protocol developed by Google.

3. List 5 difference between Browser JS(console) vs Nodejs?

|  |  |  |
| --- | --- | --- |
| S.No | Javascript | NodeJS |
| 1. | Javascript is a programming language that is used for writing scripts on the website. | NodeJS is a Javascript runtime environment. |
| 2. | Javascript can only be run in the browsers. | NodeJS code can be run outside the browser. |
| 3. | It is basically used on the client-side. | It is mostly used on the server-side. |
| 4. | Javascript is capable enough to add HTML and play with the DOM. | Nodejs does not have capability to add HTML tags. |
| 5. | Javascript can run in any browser engine as like JS core in safari and Spidermonkey in Firefox. | Nodejs can only run in V8 engine of google chrome. |
| 6. | Javascript is used in frontend development. | Nodejs is used in server-side development. |
| 7. | Some of the javascript frameworks are RamdaJS, TypedJS, etc. | Some of the Nodejs modules are Lodash, express etc. These modules are to be imported from npm. |
| 8. | It is the upgraded version of ECMA script that uses Chrome’s V8 engine written in C++. | Nodejs is written in C, C++ and Javascript. |

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

[URL](https://www.geeksforgeeks.org/url-full-form/) stands for Uniform Resource Locator. URL is the address of the website which you can find in the address bar of your web browser. It is a reference to a resource on the internet, be it images, hypertext pages, audio/video files, etc.

**Example :**

<https://practice.geeksforgeeks.org/>

**What is**[DNS](https://www.geeksforgeeks.org/domain-name-server-dns-in-application-layer/)**:**

DNS is short for Domain Name System. Like a phonebook, DNS maintains and maps the name of the website, i.e. URL, and particular IP address it links to. Every URL on the internet has a unique IP address which is of the computer which hosts the server of the website requested.

**Steps for what happens when we enter a URL :**

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.

* 1. Browser Cache
  2. Operating Systems Cache
  3. Router Cache
  4. ISP Cache
  5. 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

8.Finally, Done.

THANK YOU!