**Difference between HTTP1.1 and HTTP2**

* HTTP/2 is a **binary protocol**. This is potentially much more compressed than HTTP/1.1 and this fewer bytes on-the-wire. This means less bandwidth usage and faster load times - benefits which are amplified with many smaller requests
* HTTP/2 is **multiplexed**. This means many requests can leverage an existing connection in parallel. Coming from HTTP/1.1’s one connection per request this is a huge leap in efficiency.
* HTTP/2 requests can be **pushed.**This means requests can be pre-sent (and be ready before they've even been requested). While that creates some interesting questions (mostly revolving around caches) it's a potentially exciting area of development.
* HTTP/2 must be **encrypted.**At least for browsers. This is a valuable best practice and I'm all for seeing browser vendors flex their muscle where it benefits users.

**HTTP Version History**

Invented by Tim Berners-Lee at CERN in the years 1989–1991, HTTP (Hypertext Transfer Protocol) is the underlying communication protocol of World Wide Web. **HTTP functions as a request–response protocol in the client–server computing model.** HTTP standards are developed by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), culminating in the publication of a series of Requests for Comments (RFCs). HTTP has four versions — HTTP/0.9, HTTP/1.0, HTTP/1.1, and HTTP/2.0. Today the version in common use is HTTP/1.1 and the future will be HTTP/2.0.

## **HTTP/0.9 — the One-line Protocol**

* Initial version of HTTP — a simple client-server, request-response, telnet-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, style sheets, media)
* Methods supported: GET , HEAD , POST
* Connection nature: terminated immediately after the response

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

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.

**Difference between Browser js console vs Node js**

1. JavaScript is a simple programming language which runs in any browser JavaScript Engine. Whereas Node JS is an interpreter or running environment for a JavaScript programming language which holds a lot of excesses require libraries which can easily be accessed from JavaScript programming for better use.
2. JavaScript is normally used for any client-side activity for one web application. Activity can be addressing business validation or dynamic page display in some schedule time interval or basic Ajax call kind of task. Those are used on a maximum time for any web application. Whereas Node JS mainly used for accessing or running any operating system for non-blocking operation. An operation like create or executing shell script, or getting some specific hardware related information on one call or installed certificate details in the system or lot of define task which are non-blocking on an operating system.
3. JavaScript running in any engine like Spider monkey (Firefox), JavaScript Core (Safari), V8 (Google Chrome). So JavaScript programming is very easy to write and put any running environment means proper browser. Whereas Node JS only support V8 engine which googles chrome specific. But whether it support V8 engine, written JavaScript code can able to run in any environment. So there has no browser specific constraint on it.
4. JavaScript is normally following [Java Programming language](https://www.educba.com/java-programming-language-features/) standard. There may have some different way of writing code but at the same time, we can say it following Java Programming language standard. Whereas node JS is [written in C++](https://www.educba.com/c-plus-plus-interview-questions/), and provide V8 engine base browser JavaScript running engine which helps us to run written JavaScript program in any browser environment.
5. For accessing any operating system specific non-blocking task JavaScript has some specific object but all of them are operating system specific. An example is ActiveX Control which is only running in Windows. But Node JS is given utility to run some operating system specific non-blocking tasks from any JavaScript programming. It doesn’t have any operating system specific constant. Node JS is very much familiar to create a specific binding with the file system, and also allowing developer for reading or sometimes write on disk.

**What happens when we type a URL?**

URL 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.

**What is**DNS**:**   
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 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) connection with the server using synchronize (SYN) and acknowledge (ACK) messages.
4. Browser sends an HTTP 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 and HTML.
6. Server sends out an HTTP response along with the status of response.
7. Browser displays HTML content
8. Finally, Done.