## Difference between HTTP/1.1 and HTTP/2.0

Hypertext Transfer Protocol (HTTP) is a set of standards allowing internet users to exchange website information and fetch the required resource.

Major differences between HTTP/1.1 and HTTP/2.0 are :

* HTTP/1.1 is the 3rd version of HTTP ---- standard protocol for over 15 years---- limitation is, its use of one outstanding request per TCP connection created significant overhead, slowing down page load. HTTP/2 is derived from the SPDY(speedy) protocol as a way to improve the online experience by speeding up page loads and reducing round-trip time (RTT), especially on resource-heavy web pages.
* **HTTP/2 is binary data** whereas HTTP/1.1is of textual data which is generally less efficient across a network. HTTP/2 improved on HTTP/1.1 in a number of ways that allowed for speedier content delivery and improved the user experience.
* **HTTP/2 is multiplexed**, i.e., it can initiate multiple requests in parallel over a single TCP connection. As a result, web pages containing several elements are delivered over one TCP connection. These capabilities solve the head-of-line blocking problem in HTTP/1.1, in which a packet at the front of the line blocks others from being transmitted.
* **HTTP/2 implements server push**---the server can send assets before they are requested.
* **HTTP/2 uses pipelining** whereas HTTP/1.1 the server sends responses in the same order the requests are received.
* **HTTP/2 uses header compression** i.e the information sent with a request describes the data, its origin, type, length, how long it can be cached to reduce the overhead caused by TCP’s slow-start mechanism.
* HTTP/2 can handle elements better such as whitespace, capitalization and line endings.

## HTTP Version History

**HTTP/0.9(1991)**

* Defined as **one-liner protocol**. First version proposed by Tim Berners- Lee from CERN.
* Extremely simple consists only of **GET method,** no notation of status codes or HTTP headers.
* Limits on data transfer(1024 characters) and returns only HTML files in response.

**HTTP/1.0(1996)**

* **Building extensibility.**
* Introduces HTTP methods , HTTP status codes that line at the beginning of the response and also version information is sent within request.
* Adds HTTP headers, thus provides ability to send non-HTML files .

**HTTP/1.1(1997)**

* **Standardized protocol.**
* Adds connection reuse, pipelining of requests, content-negotiation (encoding and languages), host reader.
* Uses a single persistent TCP connection (multiple transactions over the same connection).
* Added cache control mechanism.

**HTTP/2.0(2015)**

* **SPDY –** performance improvement proposed by Google.
* Binary protocol rather than text.
* Allows Multiplexing ability and header compression.
* Allows server push (request for transaction is initiated by central server).

**HTTP/3.0(2019)**

* **QUIC-** performance improvements , upcoming proposal of using custom transport layer protocol built on top of UDP instead of UDP.

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

* After hitting the URL, browser checks cache for DNS (Domain Name System) entry to find the corresponding IP address of domain name linked to website.  
   Hence, firstly, DNS query runs in the following cache one by one to find the IP address associated with the domain name. If not found in one, then continues checking to the next until found.
  + Browser Cache
  + Operating Systems Cache
  + Router Cache
  + ISP Cache
* 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.
* If in the above steps as well, no results found, then request sends to **top or root server** of the DNS hierarchy. If we are searching IP address of the top level domain (.com,.net,.Gov,. org). It tells the resolver server to search **TLD server** (Top level domain).
* Now, resolver asks TLD server to give IP address of our domain name. TLD stores address information of domain name. It tells the resolver to ask it to **Authoritative Name server.**
* The authoritative name server is responsible for knowing everything about the domain name. Finally, resolver (ISP) gets the IP address associated with the domain name and sends it back to the browser.
* Browser initiates a **TCP (Transfer Control Protocol) connection** with the server using synchronize(SYN) and acknowledge(ACK) messages.
* Browser sends an HTTP request to the web server. **GET or POST request.**
* 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**.
* Server sends out an HTTP response along with the status of response.
* Browser displays HTML content
* Finally, the requested resource gets loaded on the user interface.

## Difference between Browser Js and Node.js

NodeJS is a javascript runtime environment based on google chrome javascript engine v8.In simple words, we can say it's just the javascript engine v8 running standalone.

BrowserJS and NodeJS differ by the following factors:

* **Application**

Browser.js is mainly used for client-side applications like validations on a web page or dynamic page display and it gets executed in the browser only while in the case of Node.js javascript code gets executed outside the browser as it is an interpreter as well as an environment for running javascript and used for server-side applications. Simply, **we can say Browser.js is used for frontend while Node.js is used for backend applications.**

* **System access**

Node.js has full system access i.e can read and write directly to the file system like any other application that also concludes that we can write complete software using Node.js while Browser.js is sandboxed for the safety purposes and have access limited to the browser. Now, that also makes it important that we should refrain from running any untrusted Javascript in Node.js.

* **Missing Objects**

In Node.js many objects are missing like-

**window object** cause it doesn’t have a window to draw anything, **location object** is related to a particular URL that means it is for page-specific, **document object** also, cause it never have to render anything on a page.

while browser.js has all these as predefined objects but the browser is missing on these- **global object** contains several functions that are not available in browsers as they are needed for server-side works only, require object which is used to include modules in the app.

* **Running Engine**

Browser.js runs in any engine like Spider monkey (Firefox), JavaScript Core (Safari), V8 (Google Chrome) accordingly to the browser while Node.js runs in a V8 engine which is mainly used by google chrome.

* **Headless**

Node.js is headless i.e without any GUI while Bowsers are not headless.

* **Modularity**

In Node.js everything is a module i.e it is mandatory to keep everything inside a module while moduling is not mandatory for browser javascript.