Difference between HTTP1.1 Vs HTTP2

HTTP stands for hypertext transfer protocol, and it is the basis for almost all web applications. More specifically, HTTP is the method computers and servers use to request and send information. For instance, when someone navigates to google.com on their laptop, their web browser sends an HTTP request to the Google servers for the content that appears on the page. Then, Google servers send HTTP responses with the text, images, and formatting that the browser displays to the user.

The first usable version of HTTP was created in 1997. Because it went through several stages of development, this first version of HTTP was called HTTP/1.1. This version is still in use on the web. In 2015, a new version of HTTP called HTTP/2 was created.

HTTP/2 solves several problems that the creators of HTTP/1.1 did not anticipate. 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.

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 – like if Bob had sent Alice a Table of Contents of his novel before sending the whole thing.

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.

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, 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

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

HTTP/2 – 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.
- 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.

List 5 difference between Browser JS console vs Node JS

1. JavaScript is a simple programming language that runs in any browser JavaScript Engine. Whereas Node JS is an interpreter or running environment for a JavaScript programming language that holds many excesses, it requires libraries that can easily be accessed from JavaScript programming for better use.
2. JavaScript is normally used for any client-side activity for one web application. An 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 for 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 creating or executing a shell script, or getting some specific hardware-related information on one call or installed certificate details in the system or a lot of define task is 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 the V8 engine, which googles chrome specific. But whether it supports the 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 standard. There may have some different way of writing code, but at the same time, we can say it following the Java Programming language standard. Whereas node JS is written in C++ and provides a V8 engine base browser JavaScript running engine, it helps us run a 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 allows the developer to read or sometimes write on disk.

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

1. You enter a URL into a web browser
2. The browser looks up the IP address for the domain name via DNS
3. The browser sends a HTTP request to the server
4. The server sends back a HTTP response
5. The browser begins rendering the HTML
6. The browser sends requests for additional objects embedded in HTML (images, CSS, JavaScript) and repeats steps 3-5.
7. Once the page is loaded, the browser sends further async requests as needed.