## Difference between HTTP /1.1 and HTTP/2.0

The main difference in HTTP/1.1 and HTTP/2.0 are

* Multiplexing
* Header compression
* Server push
* Binary protocol

**Multiplexing**

**HTTP/1.1:** loads resources one after another, so if one resource cannot be loaded, it blocks all other resources behind it.

**HTTP/2.0:** 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/2.0 does this by splitting data in-to binary code messages and numbering these messages so that the client knows which stream each binary messages belong to.

**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.

**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.

**Binary protocol:**

Binary protocols consume less bandwidth, are more efficiently parsed and are less error-prone than the textual protocols used by HTTP/1.1. Additionally, they can better handle elements such as whitespace, capitalization and line endings.

## http version history

**HTTP** (Hyper Text Transfer Protocol) is the underlying protocol of the World Wide Web. Developed by Tim Berners-Lee and his team between 1989-1991, HTTP has seen many changes, keeping most of the simplicity and further shaping its flexibility

* HTTP/0.9 – One-line protocol
* HTTP/1.0 – Building extensibility
* HTTP/1.1 – The Standardized protocol
* HTTP/2.0 - Protocol for greater performance.

**HTTP/0.9:** 1991

HTTP/0.9 is extremely simple: requests consist of a single line and start with the only possible method [**GET**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET) followed by the path to the resource (not the URL as both the protocol, server, and port are unnecessary once connected to the server).

**HTTP/1.0:** 1996

HTTP/0.9 was very limited and both browsers and servers quickly extended it to be more versatile:

* 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 (like in updating or using its local cache in a specific way)
* The notion of HTTP headers has been introduced, both for the requests and the responses, allowing metadata to be transmitted and making the protocol extremely flexible and extensible.
* With the help of the new HTTP headers, the ability to transmit other documents than plain HTML files has been added (thanks to the [Content-Type](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type) header).

**HTTP/1.1:** 1997

HTTP/1.1 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.
* Chunked responses are now also supported.
* Additional cache control mechanisms have been introduced.
* Content negotiation, including language, encoding, or type, has been introduced, and allows a client and a server to agree on the most adequate content to exchange.
* Thanks to the [Host](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Host) header, the ability to host different domains at the same IP address now allows server colocation.

**HTTP/2:** May 2015

Web pages have become much more complex, even becoming applications in their own right.  HTTP/1.1 connections need requests sent in the correct order. Theoretically, several parallel connections could be used, bringing considerable overhead and complexity. For example, HTTP pipelining has emerged as a resource burden in Web development.

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.

## List 5 difference between Browser js console vs Node js

## In browser “window” is a predefined global object which has functions and attributes, where as Nodejs doesn’t have it.

## In browser “location” is another predefined object, where as Nodejs doesn’t have it.

## In browser “require” is not predefined object, where as Nodejs has it.

## In browser module is not required, where as in Nodejs you have to keep your code inside the module.

## In browser “document” is a predefined object, where as Nodejs doesn’t have it.

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

When the URL is typed in address bar in browser, the following steps are proceeded:

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