1. **Difference between HTTP1.1 vs HTTP2:-**

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.

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.

1. **History of HTTP:-**

The term [hypertext](https://en.wikipedia.org/wiki/Hypertext) was coined by [Ted Nelson](https://en.wikipedia.org/wiki/Ted_Nelson) in 1965 in the [Xanadu Project](https://en.wikipedia.org/wiki/Xanadu_Project), which was in turn inspired by [Vannevar Bush](https://en.wikipedia.org/wiki/Vannevar_Bush)'s 1930s vision of the microfilm-based information retrieval and management "[memex](https://en.wikipedia.org/wiki/Memex)" system described in his 1945 essay "[As We May Think](https://en.wikipedia.org/wiki/As_We_May_Think)". [Tim Berners-Lee](https://en.wikipedia.org/wiki/Tim_Berners-Lee) and his team at [CERN](https://en.wikipedia.org/wiki/CERN) are credited with inventing the original HTTP, along with HTML and the associated technology for a web server and a text-based web browser. Berners-Lee first proposed the "WorldWideWeb" project in 1989—now known as the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web).

The first version of the protocol had only one method, namely GET, which would request a page from a server. The response from the server was always an HTML page. The first documented version of HTTP was [HTTP V0.9](https://www.w3.org/pub/WWW/Protocols/HTTP/AsImplemented.html) (1991). [Dave Raggett](https://en.wikipedia.org/wiki/Dave_Raggett) led the HTTP Working Group (HTTP WG) in 1995 and wanted to expand the protocol with extended operations, extended negotiation, richer meta-information, tied with a security protocol which became more efficient by adding additional methods and [header fields](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields). [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [1945](https://tools.ietf.org/html/rfc1945) officially introduced and recognized HTTP V1.0 in 1996.

The HTTP WG planned to publish new standards in December 1995 and the support for pre-standard HTTP/1.1 based on the then developing [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [2068](https://tools.ietf.org/html/rfc2068) (called HTTP-NG) was rapidly adopted by the major browser developers in early 1996. End-user adoption of the new browsers was rapid. In March 1996, one web hosting company reported that over 40% of browsers in use on the Internet were HTTP 1.1 compliant. That same web hosting company reported that by June 1996, 65% of all browsers accessing their servers were HTTP/1.1 compliant.[[18]](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#cite_note-18) The HTTP/1.1 standard as defined in [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [2068](https://tools.ietf.org/html/rfc2068) was officially released in January 1997. Improvements and updates to the HTTP/1.1 standard were released under [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [2616](https://tools.ietf.org/html/rfc2616) in June 1999.

In 2007, the [HTTP Working Group](https://httpwg.org/) was formed, in part, to revise and clarify the HTTP/1.1 specification. In June 2014, the WG released an updated six-part specification obsoleting [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [2616](https://tools.ietf.org/html/rfc2616):

* [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [7230](https://tools.ietf.org/html/rfc7230), *HTTP/1.1: Message Syntax and Routing*
* [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [7231](https://tools.ietf.org/html/rfc7231), *HTTP/1.1: Semantics and Content*
* [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [7232](https://tools.ietf.org/html/rfc7232), *HTTP/1.1: Conditional Requests*
* [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [7233](https://tools.ietf.org/html/rfc7233), *HTTP/1.1: Range Requests*
* [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [7234](https://tools.ietf.org/html/rfc7234), *HTTP/1.1: Caching*
* [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [7235](https://tools.ietf.org/html/rfc7235), *HTTP/1.1: Authentication*

[HTTP/2](https://en.wikipedia.org/wiki/HTTP/2) was published as [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [7540](https://tools.ietf.org/html/rfc7540) in May 2015.

1. **Five differences between Browser JS(console) vs Nodejs:-**

* In the browser, most of the time what we are doing is interacting with the [DOM](https://flaviocopes.com/dom/), or other [Web Platform APIs](https://flaviocopes.com/web-platform/) like Cookies. Those do not exist in Node, of course. You don’t have the document, window and all the other objects that are provided by the browser.
* In the browser, we don’t have all the nice APIs that Node.js provides through its modules, like the filesystem access functionality.
* In Node.js you control the environment. Unless you are building an open source application that anyone can deploy anywhere, you know which version of Node you will run the application on. Compared to the browser environment, where you don’t get the luxury to choose what browser your visitors will use, this is very convenient. This means that you can write all the modern [ES6-7-8-9](https://flaviocopes.com/ecmascript/) JavaScript that your Node version supports.
* In browser JS we can use Babel to transform our code to be ES5-compatible before shipping it to the browser, but in Node, we won’t need that.
* Node uses the [Common JS module system](https://flaviocopes.com/commonjs/), while in the browser we are starting to see the [ES Modules](https://flaviocopes.com/es-modules/) standard being implemented.

1. **The following steps are followed by a browser when you type a URL in the address bar of a browser:-**

# The browser checks the cache for a DNS record to find the corresponding IP address of maps.google.com.

# If the requested URL is not in the cache, ISP’s DNS server initiates a DNS query to find the IP address of the server that hosts maps.google.com

# The browser initiates a TCP connection with the server.

# The browser sends an HTTP request to the webserver.

# The server handles the request and sends back a response.

# The server sends out an HTTP response.

# The browser displays the HTML content (for HTML responses, which is the most common)