**1.HTTP1.1 vs HTTP2 2. http version**

# HTTP/1.1

Developed by Timothy Berners-Lee in 1989 as a communication standard for the World Wide Web, HTTP is a top-level application protocol that exchanges information between a client computer and a local or remote web server. In this process, a client sends a text-based request to a server by calling a *method* like GET or POST. In response, the server sends a resource like an HTML page back to the client.

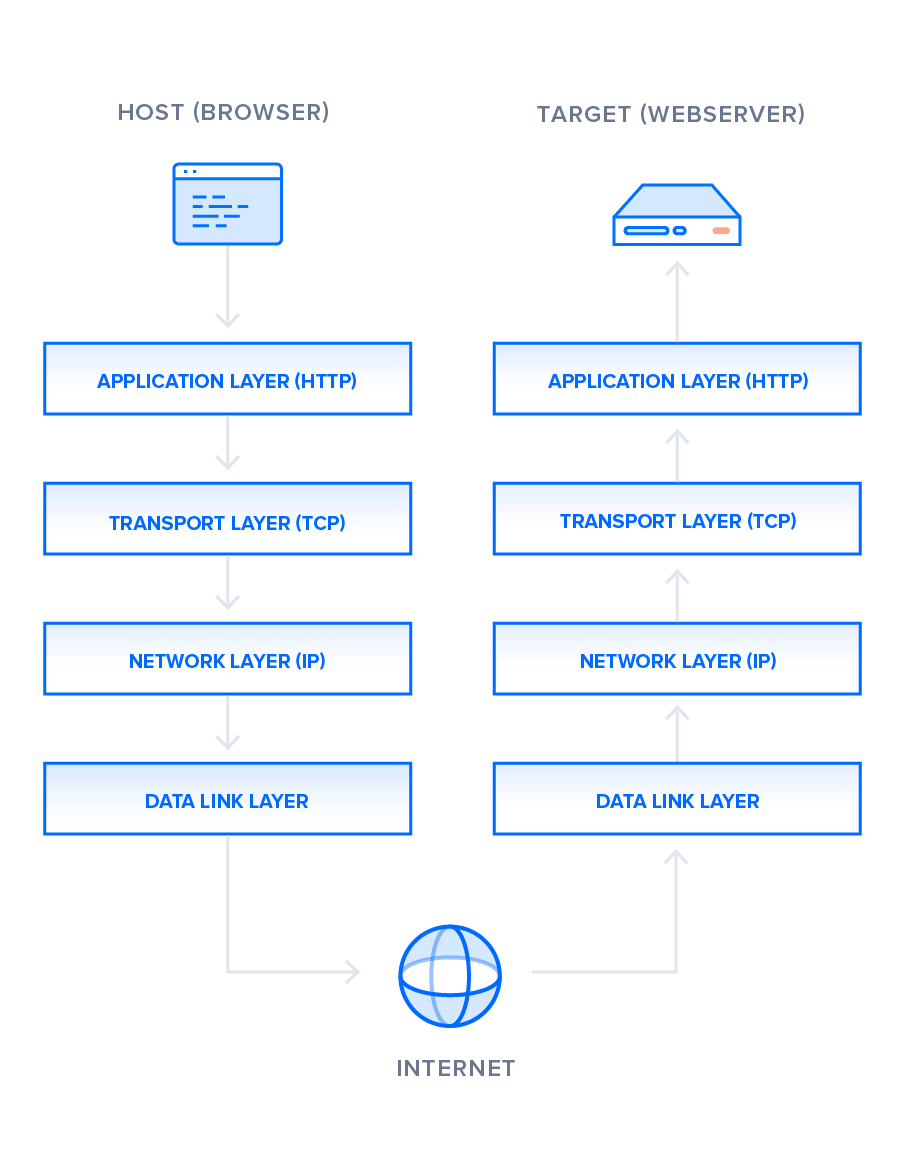
For example, let’s say you are visiting a website at the domain www.example.com. When you navigate to this URL, the web browser on your computer sends an HTTP request in the form of a text-based message, similar to the one shown here:

GET /index.html HTTP/1.1

Host: www.example.com

This request uses the GET method, which asks for data from the host server listed after Host:. In response to this request, the example.com web server returns an HTML page to the requesting client, in addition to any images, style sheets, or other resources called for in the HTML. Note that not all of the resources are returned to the client in the first call for data. The requests and responses will go back and forth between the server and client until the web browser has received all the resources necessary to render the contents of the HTML page on your screen.

You can think of this exchange of requests and responses as a single *application layer* of the internet protocol stack, sitting on top of the *transfer layer* (usually using the Transmission Control Protocol, or TCP) and *networking layers* (using the Internet Protocol, or IP)

****

There is much to discuss about the lower levels of this stack, but in order to gain a high-level understanding of HTTP/2, you only need to know this abstracted layer model and where HTTP figures into it.

With this basic overview of HTTP/1.1 out of the way, we can now move on to recounting the early development of HTTP/2.

# HTTP/2

HTTP/2 began as the SPDY protocol, developed primarily at Google with the intention of reducing web page load latency by using techniques such as compression, multiplexing, and prioritization. This protocol served as a template for HTTP/2 when the Hypertext Transfer Protocol working group http is of the [IETF (Internet Engineering Task Force)](https://www.ietf.org/) put the standard together, culminating in the publication of HTTP/2 in May 2015. From the beginning, many browsers supported this standardization effort, including Chrome, Opera, Internet Explorer, and Safari. Due in part to this browser support, there has been a significant adoption rate of the protocol since 2015, with especially high rates among new sites.

From a technical point of view, one of the most significant features that distinguishes HTTP/1.1 and HTTP/2 is the binary framing layer, which can be thought of as a part of the application layer in the internet protocol stack. As opposed to HTTP/1.1, which keeps all requests and responses in plain text format, HTTP/2 uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics, such as verbs, methods, and headers. An application level API would still create messages in the conventional HTTP formats, but the underlying layer would then convert these messages into binary. This ensures that web applications created before HTTP/2 can continue functioning as normal when interacting with the new protocol.

The conversion of messages into binary allows HTTP/2 to try new approaches to data delivery not available in HTTP/1.1, a contrast that is at the root of the practical differences between the two protocols. The next section will take a look at the delivery model of HTTP/1.1, followed by what new models are made possible by HTTP/2.

**HISTORY OF HTTP VERSIONS**

**HTTP** (HyperText 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 has evolved from an early protocol to exchange files in a semi-trusted laboratory environment, to the modern maze of the Internet, now carrying images, videos in high resolution and 3D.

## [Invention of the World Wide Web](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#invention_of_the_world_wide_web)

In 1989, while he was working at CERN, Tim Berners-Lee wrote a proposal to build a hypertext system over the Internet. Initially calling it the Mesh, it was later renamed to World Wide Web during its implementation in 1990. Built over the existing TCP and IP protocols, it consisted of 4 building blocks:

* A textual format to represent hypertext documents, the *[HyperText Markup Language](https://developer.mozilla.org/en-US/docs/Web/HTML)* (HTML).
* A simple protocol to exchange these documents, the HypertText Transfer Protocol (HTTP).
* A client to display (and accidentally edit) these documents, the first Web browser called World Wide Web.
* A server to give access to the document, an early version of http

These four building blocks were completed by the end of 1990, and the first servers were already running outside of CERN by early 1991. On August 6th 1991, Tim Berners-Lee's [post](https://www.w3.org/People/Berners-Lee/1991/08/art-6484.txt) on the public alt.hypertext  news group is now considered as the official start of the World Wide Web as a public project.

The HTTP protocol used in those early phases was very simple, later dubbed HTTP/0.9, and sometimes as the one-line protocol.

|  |  |  |
| --- | --- | --- |
| s.no | Javascript | NodeJS |
| 1. | Javascript is a programming language that is used for writing scripts on the website | NodeJS is a Javascript runtime environment. |
| 2. | Javascript can only be run in the browsers. | NodeJS code can be run outside the browser. |
| 3. | It is basically used on the client-side. | It is mostly used on the server-side. |
| 4. | Javascript is capable enough to add HTML and play with the DOM. | Nodejs does not have capability to add HTML tags. |
| 5. | It is the upgraded version of ECMA script that uses Chrome’s V8 engine written in C++. | Nodejs is written in C, C++ and  Javascript. |

# 3. difference between Browser JS(console) vs Nodejs

**4**. **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.

## Example :

https://practice.assignment.org/

#### 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.