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

**HTTP1:**

Previously, [HTTP/1.1](https://en.wikipedia.org/wiki/HTTP/2#Differences_from_HTTP_1.1) was the major version of HTTP network protocol used by the World Wide Web, implemented across clients and servers.

That worked well for 15 years. But as modern day applications and websites evolved and the amount of data to be loaded on a single page increased, the shortcomings of HTTP/1.1 became more prominent.

#### One Open Request Per Connection

HTTP/1.1 practically allows only one outstanding request per TCP connection (though [HTTP pipelining](https://en.wikipedia.org/wiki/HTTP_pipelining) allows more than one outstanding request, it still doesn’t solve the problem completely).

#### Duplication Of Data

The other problem with HTTP/1.1 is the duplication of data across requests (cookies and other headers). Too many requests means too much redundant data, which would impact performance.

**HTTP2:**

[HTTP/2](https://http2.github.io/faq/) was built over Google’s SPDY protocol with the above shortcomings of HTTP/1.1 kept in mind.

#### **Multiplexed, instead of ordered**

Allows using same TCP connection for multiple parallel requests.

#### **Header compression using HPACK**

Compressed headers, reduced data redundancy.

#### **Server Push**

Instead of waiting for the client to request for assets like JS and CSS, the server can “push” the resources it believes would be required by the client. Avoids the round trip.

**2) http version history.**

Development of HTTP was initiated by [Tim Berners-Lee](https://en.wikipedia.org/wiki/Tim_Berners-Lee) at [CERN](https://en.wikipedia.org/wiki/CERN) in 1989. Development of early HTTP [Requests for Comments](https://en.wikipedia.org/wiki/Requests_for_Comments)  was a coordinated effort by the [Internet Engineering Task Force](https://en.wikipedia.org/wiki/Internet_Engineering_Task_Force) (IETF) and the [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) with work later moving to the IETF.

HTTP/1.1 was first documented in [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [2068](https://tools.ietf.org/html/rfc2068) in 1997, and as of 2021, it (plus older versions) is less popular (used by less than a third of [websites](https://en.wikipedia.org/wiki/Website); it's always a backup protocol) for web serving than its successors. That specification was obsolete by [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [2616](https://tools.ietf.org/html/rfc2616) in 1999, which was likewise replaced by the [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [7230](https://tools.ietf.org/html/rfc7230) family of RFCs in 2014.

[HTTP/2](https://en.wikipedia.org/wiki/HTTP/2) is a more efficient expression of HTTP's semantics "on the wire", and was published in 2015, and is used by over 50% of websites; it is now supported by virtually all web browsers[[2]](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#cite_note-2) and major web servers over [Transport Layer Security](https://en.wikipedia.org/wiki/Transport_Layer_Security) (TLS) using an [Application-Layer Protocol Negotiation](https://en.wikipedia.org/wiki/Application-Layer_Protocol_Negotiation) (ALPN) extension[[3]](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#cite_note-rfc7301-3) where [TLS 1.2](https://en.wikipedia.org/wiki/TLS_1.2) or newer is required

[HTTP/3](https://en.wikipedia.org/wiki/HTTP/3) is the proposed successor to HTTP/2 and 2/3rd of web browser users (both on desktop and mobile) can already use HTTP/3, on the 18% of websites that already support it; uses [UDP](https://en.wikipedia.org/wiki/User_Datagram_Protocol) instead of [TCP](https://en.wikipedia.org/wiki/Transmission_Control_Protocol) for the underlying transport protocol. Like HTTP/2, it does not obsolete previous major versions of the protocol. Support for HTTP/3 was added to [Cloud flare](https://en.wikipedia.org/wiki/Cloudflare) and [Google Chrome](https://en.wikipedia.org/wiki/Google_Chrome) in September 2019 (since enabled by default),[[8]](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#cite_note-8)[[9]](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#cite_note-9) and can be enabled in the stable versions of Firefox[]](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#cite_note-10) and Safari.

**3). List 5 difference between Browser JS(console) vs Nodejs.**

|  |  |
| --- | --- |
| **JS(console)** |  |
| 1. | It is basically used on the client-side. |  |
| 2. | Javascript is capable enough to add HTML and play with the DOM. |  |
| 3. | Javascript can run in any browser engine as like JS core in safari and Spidermonkey in Firefox. |  |
| 4. | Javascript is used in frontend development. |  |
| 5. | Some of the javascript frameworks are RamdaJS, TypedJS, etc. |  |
| 6. | It is the upgraded version of ECMA script that uses Chrome’s V8 engine written in C++. |  |

**Nodejs**

|  |
| --- |
| 1.It is mostly used on the server-side. |
| 2.Nodejs does not have capability to add HTML tags. |
| 3.Nodejs can only run in V8 engine of google chrome. |
| 4.Nodejs is used in server-side development. |
| 5.Some of the Nodejs modules are Lodash, express etc. These modules are to be imported from npm. |
| 6.Nodejs is written in C, C++ and Javascript. |

**4) what happens when you type a URL in the address bar in the browser?**

# 1) The browser checks the cache for a DNS record to find the corresponding IP address . DNS(Domain Name System) is a database that maintains the name of the website (URL) and the particular IP address it links

# 2) 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

# 3) The browser initiates a TCP connection with the server

# Once the browser receives the correct IP address, it will build a connection with the server that matches the IP address to transfer information. Browsers use internet protocols to build such connections. There are several different internet protocols that can be used, but TCP is the most common protocol used for many types of HTTP requests.

# TCP connection is established for data transmission!

# 4) The browser sends an HTTP request to the webserver.

# The browser will send a GET request asking for web page. If you’re entering credentials or submitting a form, this could be a POST request. This request will also contain additional information such as browser identification (User-Agent header), types of requests that it will accept (Accept header), and connection headers asking it to keep the TCP connection alive for additional requests.

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

# 6) The server sends out an HTTP response.

● 1xx indicates an informational message only

● 2xx indicates success of some kind

● 3xx redirects the client to another URL

● 4xx indicates an error on the client’s part

● 5xx indicates an error on the server’s part

# 7) The browser displays the HTML content