1.Difference between HTTP1.1 vs HTTP2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1. HTTP/1.1, the first standardized version of HTTP, was introduced in 1997. It presented significant performance optimizations (over HTTP/0.9 and HTTP/1.0) and transformed the way requests and responses were exchanged between clients and servers. | 1 .At the beginning of 2010, Google introduced an experimental protocol, SPDY, which supported multiplexing (multiple requests/responses sent and received asynchronously over a single TCP connection) but as it gained traction IETF’s HTTP Working Group came up with HTTP/2 in 2015, which is based on the SPDY protocol. | | 2. It was no longer required for each connection to be terminated immediately after every request was served with a response; instead, with the keep-alive header, it was possible to have persistent connections. It allowed multiple requests/responses per TCP connection. | 2. HTTP 2.0 is a **binary** protocol that multiplexes numerous streams going over a single (normally TLS-encrypted) TCP connection. | | 3.The Upgrade header was used to indicate a preference from the client that made it possible to switch to a more preferred protocol if found appropriate by the server. | 3.It introduces the concept of a server push where the server anticipates the resources that will be required by the client and pushes them prior to the client making requests. The client retains the authority to deny the server push; however, in most cases, this feature adds a lot of efficiency to the process. | | 4. It supports connection reuse i.e. for every TCP connection there could be multiple requests and responses, and pipelining where the client can request several resources from the server at once. However, pipelining was hard to implement due to issues such as head-of-line blocking and was not a feasible solution. | 4. Uses multiplexing, where over a single TCP connection resources to be delivered are interleaved and arrive at the client almost at the same time. It is done using streams which can be prioritized, can have dependencies and individual flow control. It also provides a feature called server push that allows the server to send data that the client will need but has not yet requested. | | 5. Introduces a warning header field to carry additional information about the status of a message. Can define 24 status codes, error reporting is quicker and more efficient. | 5. Underlying semantics of HTTP such as headers, status codes remains the same. | | 6. It is relatively secure since it uses digest authentication, NTLM authentication. | 6. Security concerns from previous versions will continue to be seen in HTTP/2. However, it is better equipped to deal with them due to new TLS features like connection error of type Inadequate\_Security. | | 7. Expands on the caching support by using additional headers like cache-control, conditional headers like If-Match and by using entity tags | 7. HTTP/2 does not change much in terms of caching. With the server push feature if the client finds the resources are already present in the cache, it can cancel the pushed stream. | | 8. Spriting, concatenating, inlining, domain sharding are some of the optimizations used as a workaround to the ‘six connections per host’ rule | 8. Removes the need for unnecessary optimization hacks. | | | |
| 9. SSL is not required but recommended. Digest authentication used in HTTP1.1 is an improvement over HTTP1.0. HTTPS uses SSL/TLS for secure encrypted communication | 9. Though security is still not mandatory, it is mostly encrypted (though it is not enforced) since almost all clients require traffic to be encrypted. It also has some minimum standards, such as minimum key size for encryption. TLS 1.2 etc. |
| 10. Headers are sent on every request leading to a lot of duplicate data being sent uncompressed across the wire. | 10. Header compression is included by default in HTTP/2 using HPACK. |

2.Write about HTTP version history.

Ans: The Hypertext Transfer Protocol (HTTP) is an [application layer](https://en.wikipedia.org/wiki/Application_layer) protocol for distributed, collaborative, [hypermedia](https://en.wikipedia.org/wiki/Hypermedia) information systems. HTTP is the foundation of data communication for the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), where [hypertext](https://en.wikipedia.org/wiki/Hypertext) documents include [hyperlinks](https://en.wikipedia.org/wiki/Hyperlink) to other resources that the user can easily access, for example by a [mouse](https://en.wikipedia.org/wiki/Computer_mouse) click or by tapping the screen in a web browser.

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) (RFCs) 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) (W3C), 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 2020 it (and older versions) have minority use. That specification was obsoleted 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 50.0% 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 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, which is already in used by over 4% of websites; and is used by over 5% of desktop computers (enabled by default in latest [macOS](https://en.wikipedia.org/wiki/MacOS)), using [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 [Cloudflare](https://en.wikipedia.org/wiki/Cloudflare) and [Google Chrome](https://en.wikipedia.org/wiki/Google_Chrome) in September 2019, and can be enabled in the stable versions of Chrome and Firefox.

|  |  |
| --- | --- |
| **Year** | **HTTP Version** |
| 1991 | 0.9 |
| 1996 | 1.0 |
| 1997 | 1.1 |
| 2015 | 2.0 |
| Draft (2020) | 3.0 |

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

* 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.

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

Ans:

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