1. what is the diff between http 1 vs http 2

|  |  |  |  |
| --- | --- | --- | --- |
| Differentiator | HTTP/1.0 | HTTP/1.1 | HTTP/2 |
| Year | 1991 | 1997 | 2015 |
| Key Features | For every TCP connection there is only one request and one response. | 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. | 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. |
| Status Code | Can define 16 status codes; the error prompt is not specific enough. | 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. | Underlying semantics of HTTP such as headers, status codes remains the same. |
| Authentication Mechanism | Uses basic authentication scheme which is unsafe since username and passwords are transmitted in clear text or base64 encoded. | It is relatively secure since it uses digest authentication, NTLM authentication. | 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. |
| Caching | Provides support for caching via the If-Modified-Since header. | Expands on the caching support by using additional headers like cache-control, conditional headers like If-Match and by using entity tags. | 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. |
| Web Traffic | HTTP/1.1 provides faster delivery of web pages and reduces web traffic as compared to HTTP/1.0. However, TCP starts slowly and with domain sharding (resources can be downloaded simultaneously by using multiple domains), connection reuse and pipelining, there is an increased risk of network congestion. | | HTTP/2 utilizes multiplexing and server push to effectively reduce the page load time by a greater margin along with being less sensitive to network delays. |
| Header Compression | Headers are sent on every request leading to a lot of duplicate data being sent uncompressed across the wire. | | Header compression is included by default in HTTP/2 using HPACK. |
| Performance Optimization | Provides support for caching to deliver pages faster. | Spriting, concatenating, inlining, domain sharding are some of the optimizations used as a workaround to the ‘six connections per host’ rule. | Removes the need for unnecessary optimization hacks. |
| Protocol Type | Text based protocol that is in the readable form. | | It is a binary protocol (HTTP requests are sent in the form of 0s and 1s). Needs to be converted back from binary in order to read it. |
| Security | 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. | | 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. |

3.DIFFERENCE BETWEEN JAVASCRIPT AND NODE JS

| **.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. | Javascript can run in any browser engine as like JS core in safari and Spidermonkey in Firefox. | Nodejs can only run in V8 engine of google chrome. |
| 6. | Javascript is used in frontend development. | Nodejs is used in server-side development. |
| 7. | Some of the javascript frameworks are RamdaJS, TypedJS, etc. | Some of the Nodejs modules are Lodash, express etc. These modules are to be imported from npm. |
| 8. | 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. |

2. history of http version

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. 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; it is now supported by virtually all web browsers 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 use on the web (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.

WHAT HAPPENS WHEN YOU TYPE AN URL

**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)
4. connection with the server using synchronize(SYN) and acknowledge(ACK) messages.
5. Browser sends an HTTP request to the web server. GET or POST request.
6. Server on the host computer handles that request and sends back a response. It assembles a response in some format like JSON,XMLand HTML.
7. Server sends out an HTTP response along with the status of response.
8. Browser displays HTML content
9. Finally, Done.