* ***1.DIFFERENCES BETWEEN HTTP1.1 VS HTTP2***
  + These are the high-level differences between HTTP1 and HTTP2:
  + HTTP2 is binary, instead of textual
  + HTTP2 is fully multiplexed, instead of ordered and blocking
  + HTTP2 can, therefore, use one connection for parallelism
  + HTP2 uses header compression to reduce overhead
  + HTTP2 allows servers to “push” responses proactively into client caches
* ***2.HTTP HISTORY***
  + ***HTTP-*** HYPER TEXT TRANSFER PROTOCOL-underlying protocol of the www.
  + Developed by TIM BERNERS-LEE
  + Year:1989-1991
  + 1989:while working at CERN 🡪proposed to build a hypertext system called it MESH
  + 1990:renamed world wide web
  + Built over TCP and IP(consist of 4 building blocks)
    - A textual format to represent hypertext documents,(html)
    - Simple protocol to exchange these docs,(http).
    - Client to display these docs,the first www
    - Server to give access the docs,an early version of http
  + http(used earlier in those phases was very simple,later dubbed http/0.9)and sometimes as the online protocol.
* ***http0.9***
  + - Extremely simple.
    - Single line request starts only with possible method GET followed by the path to the resources
    - Ex:GET /mypages.html
  + (response is extremely too simple,only consisted of the file itself)
  + There were no HTTP headers, that is only html file could be transmitted, but no other type of documents.
  + There were no status or error codes in case of problem, a specific HTML file wa send back with the description of the problem contained in it, for human consumption.

***HTTP/1.0-BUILDING EXTENSIBLITIY***

* Versioning info is now sent within each request(http/1.0 is appended to the GET)
* Status code line is also sent at the beginning of the response,allowing the browser itself to understand the success of failure of the requests and to adapt its behavior in consequences.
* Notion of http headreers has been introduced,both for the request and response,allowing meta data to be transmitted and making the protocol extremely flexible and extensible
* With the help of new http headers, it allowed to transmit documents other than html  
  (thanks to the content-type header)

***HTTP/1.1- THE STANDARDIZED PROTOCOL***

* + The first standardized version of HTTP.HTTP/1.1 was published in early 1997,only a few months after HTTP/1.0.
  + --clarified ambiguities and introduced numerous improvement.
  + Connection can be reused. saving the time to reopen it numerous times to display the resources embedded into the single original documents.
  + Addition of pipelining ,allowing to sent second request before the answer of the first one to be transimmted fully.
  + Loweing latency communication
  + Chunked responses also now supported
  + Additional cache control mechanism
  + Content negotiation, including language, encoding or type has been introduced,and allows a client and a server to agree on the most adequate content to exchange.
  + Thanks to the **host** header, the ability to host different domains at the same IP address now allows server collocation.
  + http/1.1 was first published as RFC 2068 in January 1997.
  + HTTP/1.1 was refined over two revisions, RFC 2616 published in june 1999,and the series ofRFC 7230-RFC7235 published in june 2014 in prevision of the release of HTTP/2.this protocol has been extremenly stable over more than 15 years.

***HTTP for secure transmission:***

* The largest change to http done as early as of 1994, SSL additional encrypted transmission layer on top of the http.
* Ssl1.0 never released outside the company
* Ssl 2.0 and ssl 3.0 allowed for the creation of e-commerce web sites by encrypting and guaranteeing the authenticity of the messages exchanged between the server anc client.
* Ssl eventually became TLS, with version 1.0.1.1.1.2 and 1.3 appearing successfully to close vulnerabilities
* Need for encrypted transport layer raised.-->

***HTTP FOR COMPLEX APPS:***

* Orgin ver of TIM BERNERS for web wasn’t a read only medium,envisioned a web where people can add and move docs remotely,a kind of distributed file system.
* Around 1996, http extended to allow authoring, and a standard called WebDAV, was created.further extended for specific applications like CardDAV to handle address book entries and CardDAV to deal with calenders.
* All these \*DAV extension had a flaw, as had to implemented by the server to be used,quite complex.use on web realms stayed confidential.
  + - By 2000🡪new pattern for using http:REST(representational state transfer)
    - Accessing specific URIs with basic http/1.1 methods, allowed any web application to provide an API to allow retrieval and modification of its data without having to update the browsers or the servers.
* Need was embedded in the files served by the Web sites through standard HTTP/1.1.
* DRAWBACK: Each web sites defines its own non-standard RESTful API and has total control on it, unlkike the \*DAV extensions were clients and servers are interoperable.
* RESTful APIs became very common in the 2010s.
* Since 2005, the set of API available to Web pages greatly increased and several of these APIs created extension. Mostly new specific HTTP headers,to the HTTP protocol for specific purposes.
  + - Sever sent events where, the server can push occational messages to the browser.
    - Websockets, a new protocol that can be set up by upgrading an exisiting HTTP connection

***HTTP/2- A PROTOCOL FOR GREATER PERFORMANCE***

* HTTP PIPELINING has emerged as a resource burden in Web development.
* In the first half of the 2010s, Google demonstrated an alternative way of exchanging data between client and server, by implementing an experimental protocol SPDY.
* Defining an increase in responsiveness, and solving the problem of duplication of data transmitted, SPDY served as the foundations of the HTTP/2 protocol.
* The HTTP/2 protocol has several prime differences from the HTTP/1.1 version:
  + It is a binary protocol rather than text. It can no longer be read and created manually. Despite this hurdle, improved optimization techniques can now be implemented.
  + It is a multiplexed protocol. Parallel requests can be handled over the same connection, removing the order and blocking constraints of the HTTP/1.x protocol.
  + It compresses headers. As these are often similar among a set of requests, this removes duplication and overhead of data transmitted.
  + It allows a server to populate data in a client cache, in advance of it being required, through a mechanism called the server push.

[**Post-HTTP/2 evolution**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#post-http2_evolution)

* HTTP didn't stop evolving upon the release of HTTP/2. Like with HTTP/1.x previously, HTTP's extensibility is still being used to add new features. Notably, we can cite new extensions of the HTTP protocol appearing in 2016:
  + - Support of [Alt-Svc](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Alt-Svc) allows the dissociation of the identification and the location of a given resource, allowing for a smarter [CDN](https://developer.mozilla.org/en-US/docs/Glossary/CDN) caching mechanism.
    - The introduction of [Client-Hints](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Client-Hints) allows the browser, or client, to proactively communicate information about its requirements, or hardware constraints, to the server.
    - The introduction of security-related prefixes in the [Cookie](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cookie) header, now helps guarantee a secure cookie has not been altered.
    - This evolution of HTTP proves its extensibility and simplicity, liberating creation of many applications and compelling the adoption of the protocol. The environment in which HTTP is used today is quite different from that seen in the early 1990s. HTTP's original design proved to be a masterpiece, allowing the Web to evolve over a quarter of a century, without the need of a mutiny. By healing flaws, yet retaining the flexibility and extensibility which made HTTP such a success, the adoption of HTTP/2 hints at a bright future for the protocol.

***3.what happens when type url in address bar of browser?***

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