**Http Version History:**

The term hypertext was created by Ted Nelson in the year 1965. Tim Berners-Lee and its team at CERN are credited with inventing the original HTTP, along with HTML and the associated technology for a web server and a text-based web browser. Berners-Lee first proposed the "WorldWideWeb" project in 1989—now known as the World Wide Web. The first version of the protocol had only one method, namely GET, which would request a page from a server. The response from the server was always an HTML page. The first documented version of HTTP was HTTP version 0.9 in the year 1991. The HTTP working group wanted to expand the protocol with extended operations thus by updating HTTP V1.0 – RFC 1945 was created in the year 1996. The HTTP V 1.1 standard as defined in RFC2068 was officially release in January 1997. Improvements and updates to the HTTP/1.1 standard were released under RFC 2616 in June 1999. In 2007, the HTTP Working Group was formed, in part, to revise and clarify the HTTP/1.1 specification. In June 2014, the WG released an updated six-part specification obsoleting RFC 2616:

* RFC 7230, HTTP/1.1: Message Syntax and Routing
* RFC 7231, HTTP/1.1: Semantics and Content
* RFC 7232, HTTP/1.1: Conditional Requests
* RFC 7233, HTTP/1.1: Range Requests
* RFC 7234, HTTP/1.1: Caching
* RFC 7235, HTTP/1.1: Authentication

HTTP/2 was published as RFC 7540 in May 2015.

**HTTP V1.1 vs HTTP V2.0**

1. HTTP/2 can send multiple requests for a data in parallel over a single TCP connection. This is the most advanced feature of the HTTP/2 protocol because it allows you to download web files asynchronously from one server. Most modern browsers limit TCP connections to one server. Whereas is HTTP/1.1 the limit of the TCP connections to server exceeds one. This reduces additional round trip time (RTT), making your website load faster without any optimization, and makes domain sharding unnecessary.
2. HTTP/2 compress a large number of redundant header frames. It uses the HPACK specification as a simple and secure approach to header compression. Both client and server maintain a list of headers used in previous client-server requests.
3. The latest HTTP version has evolved significantly in terms of capabilities and attributes such as transforming from a text protocol to a binary protocol. HTTP1.x used to process text commands to complete request-response cycles. HTTP/2 will use binary commands (in 1s and 0s) to execute the same tasks. This attribute eases complications with framing and simplifies implementation of commands that were confusingly intermixed due to commands containing text and optional spaces. Because of this Less prone to errors, Lighter network footprint, Effective network resource utilization, Compact representation of commands for easier processing and implementation, Efficient and robust in terms of processing of data between client and server, Reduced network latency and improved throughput.
4. HTTP/2 Server Push capability allows the server to send additional cacheable information to the client that isn’t requested but is anticipated in future requests.
5. For HTTP/1.1 we might require a greater number of TCP connection but in HTTP/2 several request shall be placed in a Single TCP connection. Thus, making the other TCP available for other clients and reduce the load.

**What happens when you type a URL in the address bar of a browser?  
Step1:**

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

**Step2:**

If not found in cache, ISP’s DNS server initiates a DNS query to find the IP address of server that hosts the domain name. The request are sent using small data packets that contains the information content of request and IP address it is destined for.

**Step 3:**

Browser initiates a TCP connection with the server using the synchronize and acknowledge message.

**Step 4:**

Browser sends an HTTP request to the web server. GET or POST request.

**Step 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.

**Step 6:**  
Server sends out an HTTP response along with the status of response.

**Step 7:**

Browser displays HTML content

**Difference between Node.js and Browser console**

|  |  |
| --- | --- |
| **Node.js** | **Browser Console** |
| Node executes JavaScript in server side | Browser Console executes in client side |
| Node doesn't have a predefined "window" object because it doesn't have a window to draw anything. | "window" is a predefined global object which has functions and attributes, that have to deal with window that has been drawn. |
| "location" object is related to a particular url; that means it is for page specific. So, node doesn't require that. | "location" is another predefined object in browsers, that has all the information about the url we have loaded. |
| Node doesn't have "document" object also, cause it never have to render anything in a page. | "document", which is also another predefined global variable in browsers, has the html which is rendered. |
| Node is headless. | Browsers are not headless. |
| Node processes request object. | Browsers processes response objects. |
| In Node everything is a module. You must keep your code inside a module. | Moduling is not mandatory in client-side JavaScript, i.e. in browsers. |