1. **Difference between HTTP 1.1 vs HTTP 2 :**

\* Hypertext Transfer Protocol (HTTP) is an application protocol that is, currently, the foundation of data communication for the World Wide Web.

\* HTTP is based on the Client/Server model. Client/Server model can be explained as two computers, Client (receiver of service) and Server (provider of service) that are communicating via requests and responses.

\* HTTP1.1 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.

\* HTTP2 is binary HTTP1 is textual.

\* HTTP2 is fully multiplexed, HTTP 1.1 is 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.

\* HTTP2 allows to download web files asynchronously from one server.

\* The server can multiplex pushed resources along with originally requested information within the same TCP connection.

\* HTTP/2 provides us with many new mechanics that will mitigate HTTP/1.1 issues and ones that will boost your web page performance.

\* HTTP/2 can send multiple requests for data in parallel over a single TCP connection.

|  |  |
| --- | --- |
| ***HTTP/2:***   * Single Connection - One TCP connection is required to load the entire website thereby improving webpage load speed. * Multiplexing - Can send multiple parallel requests for data over a single connection. * Binary Protocol - Easier to parse and consumes less bandwidth. * Header compression - Compression of requests header reduces the overhead. * Server push - Server sends resources to client that are not requested but likely to be used in future, which are stored in cache by the client. ( i.e sending both HTML and CSS file for a HTML request) * Stream prioritization - loading various elements of the website in proper order. (i.e loading HTML and CSS files before JavaScript files) | ***HTTP/1.1***   * Multiple connections - are needed to load various elements of a webpage, hence page load speed is comparatively less. (i.e one request per TCP connection) * Pipe-lining - second request is sent before the first request is fully served. * No Header compression - many duplicate data are sent uncompressed on every request. * Text based protocol - Requests are sent in readable format unlike HTTP/2 . * No need for Stream prioritization because of head of line blocking. * No Server push - only data that are requested are sent. |

\* HTTP/1.1 and HTTP/2 protocol with a slight difference.

\* Imagine that waiters are TCP connections and you want to order your meal and a bottle of water. For HTTP/1.1 that would mean that you ask one waiter for your meal and another one for water, hence you would allocate two TCP connections. For HTTP/2 that would mean that you ask only one waiter for both, but he brings them separately. You only allocate one TCP connection and that will already result with lower server load, plus the server would have one extra free connection (waiter) for the next client (guest).

\* The real difference between HTTP/1.1 and HTTP/2 comes with **server push example**.

\* Imagine that the guest (Client) asks (sends request) waiter (Server) for a meal, then the waiter gets the meal from the restaurant chef (your application logic), but the waiter also thinks you would need a bottle of water so he brings that too with your meal. The end result of this would be only one TCP connection and only one request that will significantly lower the server load.

1. **Objects and its Internal Representation in JavaScript :**

\* JavaScript is designed on a simple object-based paradigm.

\* An object is a collection of properties, and a property is an association between a name (or key) and a value.

\* A property’s value can be a function, in which case the property is known as a method.

\* A JavaScript object has properties associated with it. A property of an object can be explained as a variable that is attached to the object.

\* Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects. The properties of an object define the characteristics of the object.

\* Unlike objects in other programming languages JavaScript has an unique approach towards objects and their representation. Objects in JavaScript are a collection of key-value pair.

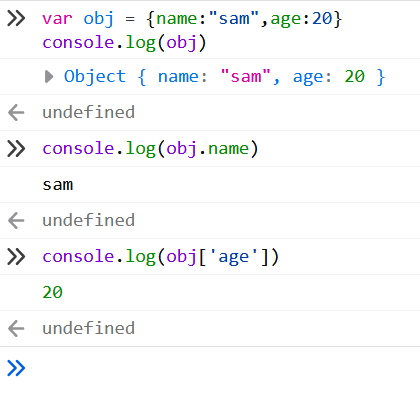
\* The collection can also include functions, which are called member functions and all the other keys are called its properties. Keys can either be a number or string.

\* Keys are used to access all the elements in an object. They can be accessed in two ways:

\* Dot notation

\* Array notation (i.e [])

Example :



\* A simple diagram is probably the best way to give a quick overview of the object representation in JavaScript.

