Assignment 1

# HTTP/1.1 vs HTTP/2

|  |  |
| --- | --- |
| HTTP/1.1 was established in 1989. | HTTP/2 was introduced in 2015. |
| Response and request messages between server and client are transferred as plain-text messages. | Response and request messages are encoded into binary and are transferred as such. |
| Multiple TCP connections are used in order to avoid head of line blocking, in which the request at the top of the queue blocks all the others behind it. | A single TCP connection with multiple streams of data is established, in which each stream is divided into even smaller frames. |
| Memory buffer control is done on the transport layer and by using system default values. Thus, flow control can only be implemented at either end of the connection. An ACK packet is used to determine the receive window i.e memory left in the buffer. Each TCP connection requires its own flow control. | Since multiple streams of data are used in only one TCP connection, flow control is transferred to the Application Layer. Hence the receive window can be modified mid-way through the transmission and intermediary nodes can modify and determine their resource allocation accordingly. |
| Resource inlining is used when the developer knows what resource a client is going to need. These resources are then included in the HTML response that the client will receive. But this can cause resource redundancy and if the resource is too big it defeats the purpose of inlining. Also the client cannot separate the resource and the HTML document. | As HTTP2 uses multiple streams of data it has the ability to send the HTML response and the resource at the same time, thereby solving the problem with resource inlining. This is done by including frames in the message which indicate if server is sending a resource. The client can accept or reject this. This is also known as server push. |

# 

# HTTP Version History

**HTTP 0.9**

The earliest version of http, it wasn’t given any version number until later on to differentiate it from the subsequent versions. Only the GET method was possible for requests. Requests consisted of a single line with the GET method followed by the path to the resource. There were no http headers and except HTML, no other type of file could be transferred. There were also no status and error codes and in case of an error a specific HTML was sent explaining the error in a human digestible manner.

**HTTP 1.0**

This version of http introduced a lot of new features to make the protocol more extensible and versatile. The concept of status code was introduced so that the browser itself can understand the success or failure of the request and respond accordingly.

The notion of HTTP headers was also introduced which allowed transfer of metadata and also allowed the transmission of other file types with the help of the “Content-Type” header.

**HTTP 1.1**

This was the first standardized version of HTTP that brought new features and clarified some ambiguities. Connection reuse was made possible after this version, allowing a TCP connection to stay open between the server and the client, saving the time to reopen to access resources embedded into a single HTML document.

Pipelining was added, which enabled a client to send consecutive requests without receiving the response for the initial request.

Content negotiation between the host and the client was introduced, enabling them to choose the most adequate content for exchange.

**HTTP 2**

Over the years after the introduction of HTTP 1.1, web pages had become much more complex and versatile. The amount of textual and visual data displayed had increased, the volume of scripts adding functionality had also seen the same rise.

HTTP2 was introduced to increase the efficiency and performance of the communication between the server and the client.

It introduced binary protocol rather than text, enabling more information to be sent with more efficiency.

Binary protocol opened the gates for multiplexing, parallel requests can now be handled on the same TCP connection without any blocking constraints.

It also introduced compression of headers similar to a set of requests, enabling redundancy management.

New ways of sending resources to clients were created, like server push.

# Browser JS vs NodeJs

The main difference between nodejs and browser js is that of the environment.

Building applications that run in the browser is a completely different thing than building a nodejs application.

**DOM and UI** **-** Unlike browser js, nodejs doesn’t have any access to DOM and UI elements, cookies etc.

**APIs and Modules -** In the same context browser js doesn’t have access to the various kinds of APIs and modules that nodejs provides, like the filesystem access functionality.

**Environment Control -**  One big plus for nodejs developers is that often they know exactly which version of nodejs are they targeting with their application. Much unlike browser js where developers have to keep in mind the compatibility between various kinds of browsers and their versions.

**Future Updates -** Javascript updates move quick and browsers are often a bit behind these updates and users more so. Nodejs as a standalone project has the advantage of being able to get up to date with newer versions of Javascript quicker than browsers.

**CommonJS -** Unlike browserjs, nodejs uses the CommonJS module system.

# What happens when you search a URL?

When we search for a url, the browser checks local cache to find the corresponding address for the domain name. If not found it goes to ask the ISP DNS about it. This is done with a DNS query.

Once the ip address is found the browser initiates a TCP connection with the host server.

The browser sends an HTTP request with GET method.

Server on the other hand handles the request and sends an HTTP response. This response contains the list of resources the client is to expect followed by the resources itself. This response also contains status code which would indicate success or failure of request.

Browser then displays the HTML content received.