**Name**: Ashokkumar.R

**Batch: b24** weekday

**Sub**:java script

**Email id**: [ashokkumar.rmca@gmail.com](mailto:ashokkumar.rmca@gmail.com)

**Date**:11/05/2021

**Task1**

**1)**HTTP1.1 vs HTTP 2

HTTP1.1

HTTP:

* Hyper transfer text protocol
* stands for hypertext transfer protocol, and it is the basis for almost all web applications.
* More specifically, HTTP is the method computers and servers use to request and send information
* For instance, when someone navigates to cloudflare.com on their laptop, their web browser sends an HTTP request to the Cloudflare servers for the content that appears on the page. Then, Cloudflare servers send HTTP responses with the text, images, and formatting that the browser displays to the user.
* For instance, when someone navigates to cloudflare.com on their laptop, their web browser sends an HTTP request to the Cloudflare servers for the content that appears on the page. Then, Cloudflare servers send HTTP responses with the text, images, and formatting that the browser displays to the user.

|  |  |  |
| --- | --- | --- |
| NAME | HTTP1.1 | HTTP2.0 |
| Solving capacity | Less | more |
| speed | low | Very high |
| Efficient | less | More |
| Prioritization | Not there | There |
| Developer | Not hands on by developers | Very much detail handled by developers hands on |
| Send capacity | One at a time | More at a time |
| Time | So more time taken | Less time taken |
| Weighted prioritization | Not there | Its there so multitasking can be used |
| Displaying fecility | Not there | It can decide which page to display first. Like html or CSS or js which content |
| Multiplexing | Resources one after another. So block will happen | Send it in a single tcp connection. So receives multiple by binary code. No blocks |
| Server push | Client have to ask or wait for reply | Here it is easy . before client ask it would come. By the method of server push |
| Header compression | To view the images faster manner | Here very fast ,compression method called HPACK will be used it is advanced method . |

**Evolution of HTTP**

**HTTP** (HyperText Transfer Protocol) is the underlying protocol of the World Wide Web. Developed by Tim Berners-Lee and his team between 1989-1991, HTTP has seen many changes, keeping most of the simplicity and further shaping its flexibility. HTTP has evolved from an early protocol to exchange files in a semi-trusted laboratory environment, to the modern maze of the Internet, now carrying images, videos in high resolution and 3D.

initially calling it the Mesh, it was later renamed to World Wide Web during its implementation in 1990. Built over the existing TCP and IP protocols, it consisted of 4 building blocks:

* A textual format to represent hypertext documents, the [*HyperText Markup Language*](https://developer.mozilla.org/en-US/docs/Web/HTML) (HTML).
* A simple protocol to exchange these documents, the HypertText Transfer Protocol (HTTP)
* A client to display (and accidentally edit) these documents, the first Web browser called *WorldWideWeb*.
* A server to give access to the document, an early version of *httpd*.

## [HTTP/0.9 – The one-line protocol](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#http0.9_%E2%80%93_the_one-line_protocol)

The initial version of HTTP had no version number; it has been later called 0.9 to differentiate it from the later versions. HTTP/0.9 is extremely simple: requests consist of a single line and start with the only possible method [GET](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET) followed by the path to the resource (not the URL as both the protocol, server, and port are unnecessary once connected to the server).

<HTML>

A very simple HTML page

</HTML>

Unlike subsequent evolutions, there were no HTTP headers, meaning that only HTML files could be transmitted, but no other type of documents. There were no status or error codes: in case of a problem, a specific HTML file was send back with the description of the problem contained in it, for human consumption.

## [**HTTP/1.0 – Building extensibility**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#http1.0_%E2%80%93_building_extensibility)

HTTP/0.9 was very limited and both browsers and servers quickly extended it to be more versatile:

* Versioning information is now sent within each request (HTTP/1.0 is appended to the GET line)
* A status code line is also sent at the beginning of the response, allowing the browser itself to understand the success or failure of the request and to adapt its behavior in consequence (like in updating or using its local cache in a specific way)
* The notion of HTTP headers has been introduced, both for the requests and the responses, allowing metadata to be transmitted and making the protocol extremely flexible and extensible.
* With the help of the new HTTP headers, the ability to transmit other documents than plain HTML files has been added (thanks to the [Content-Type](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type) header).

These novelties have not been introduced as concerted effort, but as a try-and-see approach over the 1991-1995 period: a server and a browser added one feature and it saw if it got traction. A lot of interoperability problems were common. In November 1996, in order to solve these annoyances, an informational document describing the common practices has been published, [RFC 1945](https://tools.ietf.org/html/rfc1945). This is the definition of HTTP/1.0 and it is notable that, in the narrow sense of the term, it isn't an official standard.

## [HTTP/1.1 – The standardized protocol](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#http1.1_%E2%80%93_the_standardized_protocol)

In parallel to the somewhat chaotic use of the diverse implementations of HTTP/1.0, and since 1995, well before the publication of HTTP/1.0 document the next year, proper standardization was in progress. The first standardized version of HTTP, HTTP/1.1 was published in early 1997, only a few months after HTTP/1.0.

HTTP/1.1 clarified ambiguities and introduced numerous improvements:

* A connection can be reused, saving the time to reopen it numerous times to display the resources embedded into the single original document retrieved.
* Pipelining has been added, allowing to send a second request before the answer for the first one is fully transmitted, lowering the latency of the communication.
* Chunked responses are now also supported.
* Additional cache control mechanisms have been introduced.
* 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](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Host) header, the ability to host different domains at the same IP address now allows server colocation.

## [More than 15 years of extensions](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#more_than_15_years_of_extensions)

prevision of the release of HTTP/2, this protocol has been extremely stable over more than 15 years.

The largest change that happened to HTTP was done as early as end of 1994. Instead of sending HTTP over a basic TCP/IP stack, Netscape Communications created an additional encrypted transmission layer on top of it: SSL. SSL 1.0 was never released outside the company, but SSL 2.0 and its successor 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 and client. SSL was put on the standards track and eventually became TLS, with versions 1.0, 1.1, 1.2, and 1.3 appearing successfully to close vulnerabilities.

USING FOR COMPLEX APPLICATIONS

HTTP has been extended to allow authoring, and a standard called WebDAV was created. It has been further extended for specific applications like CardDAV to handle address book entries and CalDAV to deal with calendars. But all these \*DAV extensions had a flaw: they had to be implemented by the servers to be used, which was quite complex. Their use on Web realms stayed confidential.

REST model

This 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: all what is needed was embedded in the files served by the Web sites through standard HTTP/1.1. The drawback of the REST model resides in the fact that each website defines its own non-standard RESTful API and has total control on it; unlike the \*DAV extensions where clients and servers are interoperable. RESTful APIs became very common in the 2010s.

## [HTTP/2 – A protocol for greater performance](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#http2_%E2%80%93_a_protocol_for_greater_performance)

* Speed
* Performance
* Multiplexing
* More clarity
* Priority
* Header
* Server push

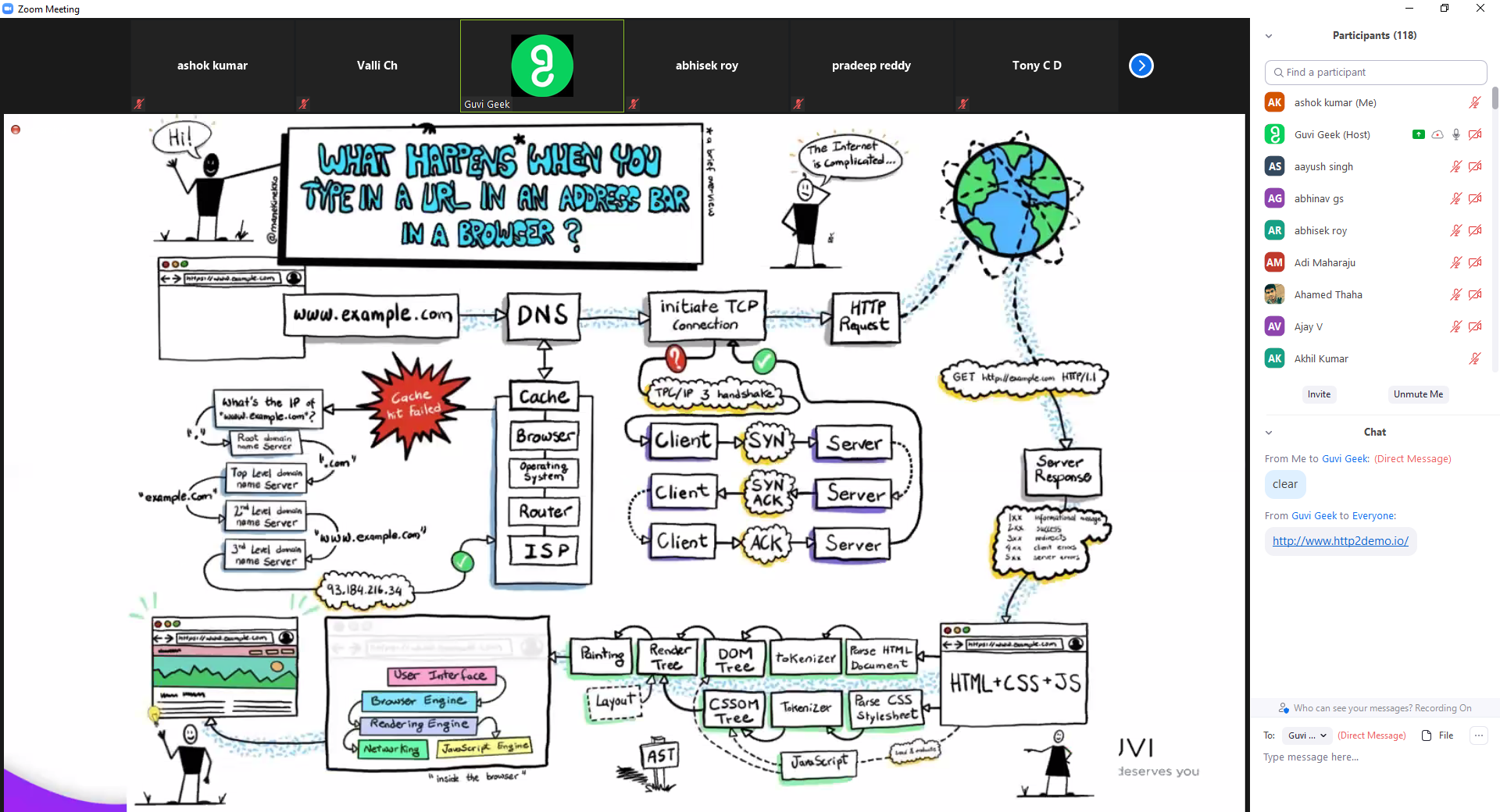
**HTTP/3 - HTTP over QUIC**

The next major version of HTTP, HTTP/3, will use QUIC instead TCP/TLS for the transport layer portion.

3)JS CONSOLE vs NODE JS

|  |  |  |
| --- | --- | --- |
| N0 | Java script | Node js |
| 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 Spider monkey 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. |
| 9 | It doesn’t control the environment | Controls the environment |

4)what happens when you type the url in addres bar in a browser ??



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