**TASK :1**

**1.Difference between HTTP/1.1 – HTTP/2**

HTTP stands for Hypertext Transfer Protocol. Its used in client - server communication. By using HTTP the user send the request to server and server send response to the user. There are several stages of development of HTTP. HTTP 1.1 is developed in 1997 and HTTP 2 is developed in 2015.

HTTP/1.1:

For better understanding , lets assume that we send a request to the server for www.google.com page & the server responds you as a resource [www.google.com](http://www.google.com) page . Before sending request to server and server response there is a TCP(Transmission control Protocol) established a connection between server and user. Again we send a request to another page or format( image.img.jpg) the server gives response to that format resource file . The connection between user &server is not break here after the first request because we add a keep-alive header which is the part of the request, so there is a open connection between user & server. There is a persistent connection which means several Request & response are merged together in a single connection. HTTP/1.1 transfer all the request & response in a plain text message form , and Head of line blocking in all which TCP connection is blocked all other request until the response does not receive, all the information or resource related to the header file is repeated in every request. These are the drawbacks that leads to the creation of HTTP/2:

HTTP/2:

HTTP/2: was developed over the SPDY Protocol ( an Application Layer protocol for transporting content over the web, designed specifically for minimal latency). HTTP/2 works on the binary framing layer instead of textual that converts all the messages into Binary format. Its works on multiplexed that is one TCP used for multiple requests. HTTP/2 used HPACK which is used to split data from header and compressed the header. The server sends all other files like CSS & JS without the request of the client using the PUSH frame.

Some Difference between the HTTP/1.1: and HTTP/2: are,

|  |  |
| --- | --- |
| HTTP/1.1 | HTTP/2 |
| It works on the textual format | It works on the Binary Protocol |
| There is head of line blocking that blocks all the request behind it until its doesn’t get its all resources. | Its allows multiplexing , so one TCP connection is requires for multiple requests |
| Its uses request resources inlining for use of getting multiple pages. | Its use PUSH frame by server that collects all multiple pages. |
| It compress data by itself | Its uses HPACK for data compression |

**2. Objects and its internal representations of JavaScript ;**

Objects, in JavaScript, is it’s most important data-type and forms the building blocks for modern JavaScript. These objects are quite different from JavaScript’s primitive data-types(Number, String, Boolean, null, undefined and symbol) in the sense that while these primitive data-types all store a single value each. Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types. objects in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of “key: value” pairs. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object. Every object has some property associated with some value. These values can be accessed using these properties associated with them.

Programming an object involves declaring **instance variables** to represent the object’s state, and writing **instance methods** that implement behavior. The separate parts of an object’s state are often called its **properties**. Thus, you might have a bank account object which had properties such as holder of account, address of holder, balance and credit limit; its methods would include the likes of credit, debit and change address. JavaScript represents functions as data of type Function, so in fact there is less difference between **data properties** and **functions** than in most other object-based programming languages. Functions can be passed and stored in variables, and any function stored in a particular variable can be executed when required.

**3. Read about IP address, port, HTTP methods, and MAC address**

**IP Address:**

An IP address is analogous to a street address or telephone number in that it is used to uniquely identify a device on the Internet to deliver email message, or visit ("view") domain names. Your IP address is usually based on a real-world location, and this website will use your IP address to guess where you are (which country you are from). An **Internet Protocol address** (**IP address**) is a numerical label such as *192.0.2.1* that is connected to computer network that uses the Internal Protocol for communication. An IP address serves two main functions: network interface identification, and location addressing.

**HTTP Methods:**

HTTP defines a set of **request methods** to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred to as HTTP verbs. Each of them implements a different semantic, but some common features are shared by a group of them: e.g., a request method can be safe, idempotent, or catchable.

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

[HEAD](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD)

The HEAD method asks for a response identical to a GET request, but without the response body.

[POST](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST)

The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.

[PUT](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT)

The PUT method replaces all current representations of the target resource with the request payload.

[DELETE](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE)

The DELETE method deletes the specified resource.

[CONNECT](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/CONNECT)

The CONNECT method establishes a tunnel to the server identified by the target resource.

[OPTIONS](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS)

The OPTIONS method describes the communication options for the target resource.

[TRACE](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/TRACE)

The TRACE method performs a message loop-back test along the path to the target resource.

[PATCH](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH)

The PATCH method applies partial modifications to a resource.

**PORT:**

A port is a virtual point where network connections start and end. Ports are software-based and managed by a computer's operating system. Each port is associated with a specific process or service. Ports allow computers to easily differentiate between different kinds of traffic: emails go to a different port than webpages, for instance, even though both reach a computer over the same Internet connection. Ports are standardized across all network-connected devices, with each port assigned a number. Most ports are reserved for certain protocols for example, all Hypertext Transfer Protocol (HTTP) messages go to port 80. While IP address enable messages to go to and from specific devices, port numbers allow targeting of specific services or applications within those devices.

**MAC address:**

A **MAC address** short for **Media Access Control address** is a Unique Identifier assigned to a  Network Interface Controller (NIC) for use as a network address in communications within a network segment. This use is common in most IEEE 802 networking technologies, including Ethernet, Wi-Fi, and Bluetooth. Within the Open System Interconnection, MAC addresses are used in the medium access control protocol sublayer of the data link layer. As typically represented, MAC addresses are recognizable as six groups of two hexadecimal digits, separated by hyphens, colons, or without a separator. MAC addresses are primarily assigned by device manufacturers, and are therefore often referred to as the **burned-in address**, or as an **Ethernet hardware address**, **hardware address**, or **physical address**. Each address can be stored in hardware, such as the card's read-only memory, or by a firmware mechanism. Many network interfaces, however, support changing their MAC address. The address typically includes a manufacturer's organizationally unique identifier (OUI). MAC addresses are formed according to the principles of two numbering spaces based on extended unique identifiers (EUIs) managed by the Institute of Electrical and Electronics Engineers (IEEE): **EUI-48**—which replaces the obsolete term **MAC-48**—and **EUI-64**.

Network nodes with multiple network interfaces, such as routers and Multilayer Switches, must have a unique MAC address for each NIC in the same network. However, two NICs connected to two different networks can share the same MAC address.

MAC addresses work with the card in your device that lets it connect wirelessly to the internet, called a Network Interface Controller (NIC). MAC addresses are used to identify which device is which on your local network so that data gets sent to your computer and not your roommate's smartphone. MAC addresses are always a 12 digit hexadecimal number, with the numbers separated every two digits by a colon or hyphen. So a MAC address of 2c549188c9e3, for example, would be displayed 2C:54:91:88:C9:E3 or 2c-54-91-88-c9-e3.