**Day 01: Tasks**

1. Difference between HTTP1.1 Vs. HTTP2:

**HTTP1.1**: It was introduced in the year 1997.

Key Features: It supports connection reuse i.e for every TCP connection there could be multiple requests and responses, and pipelining where the client can request several resources from the server at once. However, pipelining was hard to implement due to issues such as head-of-line blocking and was not a feasible solutions.

Status Code: Introduces a warning header field to carry additional information about the status of a message. Can define 24 status codes, error reporting is quicker and more efficient.

Authentication Mechanism: It is relatively secure since it uses digest authentication, NTLM authentication.

Caching: Expands on the caching support by using additional headers like cache-control, conditional headers like If-Match and by using entity tags.

Web Traffic: HTTP1.1 provides faster delivery of web pages and reduces web traffic as compared to HTTP1.0. However, TCP starts slowly and with domain sharing (resources can be downloaded simultaneously by using multiple domains), connection reuse and pipelining, there is an increased risk of network congestion.

**HTTP2.0:** It was introduced in the year 2015.

Key Features: Uses multiplexing, where over a single TCP connection resources to be delivered are interleaved and arrive at the client almost at the same time. It is done using streams which can be prioritized, can have dependencies and individual flow control. It also provides a feature called server push that allows the server to send data that the client will need but has not yet requested.

Status Code: Underlying semantics of HTTP such as headers, status codes remains the same.

Authentication Mechanism: Security concerns from previous versions will continue to be seen in HTTP2. However, it is better equipped to deal with them due to new TLS features like connection error of type (Inadequate\_Security).

Caching: HTTP2 does not change much in terms of caching. With the server push feature if the client finds the resources are already present in the cache, it can cancel the pushed stream.

Web Traffic: HTTP2 utilizes multiplexing and server push to effectively reduce the page load time by a greater margin along with being less sensitive to network delays objects and its internal representation in Javascript.

1. Objects and its internal representations in Javascript:

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. You access the properties of an object with a simple dot-notation:

objectName.propertyName

Like all JavaScript variables, both the object name (which could be a normal variable) and property name are case sensitive. You can define a property by assigning it a value. For example, let’s create an object named myCar and give it properties named make, model and year as follows:

var myCar = new Object();  
myCar.make = 'Ford';  
myCar.model = 'Mustang';  
myCar.year = 1969;

Unassigned properties of an object are undefined (and not null).

myCar.color; // undefined

Properties of JavaScript objects can also be accessed or set using a bracket notation. Objects are sometimes called associative arrays, since each property is associated with a string value that can be used to access it. So, for example, access the properties of the myCar object as follows:

myCar['make'] = 'Ford';  
myCar['model'] = 'Mustang';  
myCar['year'] = 1969;

We can also access properties by using a string value that is stored in a variable:

var propertyName = 'make';  
myCar[propertyName] = 'Ford';propertyName = 'model';  
myCar[propertyName] = 'Mustang';

The following function displays the properties of the object when you pass the object and the object's name as arguments to the function:

function showProps(obj, objName) {  
var result = ``;  
for (var i in obj) {  
// obj.hasOwnProperty() is used to filter out properties from the object's prototype chain  
if (obj.hasOwnProperty(i)) {  
result += `${objName}.${i} = ${obj[i]}\n`;  
}  
}  
return result;  
}

So, the function call showProps(myCar, "myCar") would return the following:

myCar.make = Ford  
myCar.model = Mustang  
myCar.year = 1969