**Task 1**

Do a write up for the followings:

**1.Difference between HTTP1.1 vs HTTP2**

* **HTTP1.1** keeps all requests and responses in plain text format while **HTTP2** uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics, such as verbs, methods, and headers.
* Improved delivery models ( data delivery ) with **HTTP2** over **HTTP1.1,** since binary framing layer improves speed and flexibility of data transfer.
* **HTTP 2.0** is a binary protocol that multiplexes numerous streams going over a single (normally TLS-encrypted) TCP connection

**2.HTTP version history**

**HTTP/0.9 - 1991** — The One-line Protocol

* Initial version of HTTP — a simple client-server, request-response, telenet-friendly protocol
* Request nature: single-line (method + path for requested document)
* Methods supported: GET only
* Response type: hypertext only
* Connection nature: terminated immediately after the response
* No HTTP headers (cannot transfer other content type files), No status/error codes, No URLs, No versioning

**HTTP/1.0 - 1996** — Building extensibility

* Browser-friendly protocol
* Provided header fields including rich metadata about both request and response (HTTP version number, status code, content type)
* Response: not limited to hypertext (Content-Type header provided ability to transmit files other than plain HTML files — e.g. scripts, stylesheets, media)
* Methods supported: GET , HEAD , POST
* Connection nature: terminated immediately after the response

**HTTP/1.1- 1997** — The standardized protocol

* This is the HTTP version currently in common use.
* Introduced critical performance optimizations and feature enhancements — persistent and pipelined connections, chunked transfers, compression/decompression, content negotiations, virtual hosting (a server with a single IP Address hosting multiple domains), faster response and great bandwidth savings by adding cache support.
* Methods supported: GET , HEAD , POST , PUT , DELETE , TRACE , OPTIONS
* Connection nature: long-lived

**HTTP2.0 - 2015**- released by Internet Engineering Task Force aimed at improving overall HTTP performance.

**HTTP3.0 - 2020** - HTTP 3 intends to simplify the TCP handshake. the connections are established faster using the QUIC protocol, which is something new as well.

**3.List 5 differences between Browser JS vs Node Js.**

Browser JS:

* Used on client side
* Can use HTML tags - DOM, web API, cookies
* Supports multiple browser engines
* Run time environment in web browser.
* ES standard modules - **Import**

Node JS:

* Server side
* Run time outside of browser
* Can’t use HTML tags
* Can run only on V8 engine of google chrome.
* Interacts using file system access functionality
* Common JS module system - uses **require()**

**4.what happens when you type a URL in the address bar in the browser?**

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

Task2

**1.Print all country names in console.**

Html file

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>api data</title>

</head>

<body>

    <script src ='script.js'></script>

</body>

</html>

Script.js

var xmlhttp = new XMLHttpRequest();

xmlhttp.open('GET','https://restcountries.eu/rest/v2/all', true);

xmlhttp.send();

xmlhttp.onload = function() {

    var arr = JSON.parse(this.response);

    for(var i in arr)

    {

     console.log(arr[i].name);

    }

}

**2.Difference between copy by value and copy by reference.**

**Copy by value:**

In a primitive data-type when a variable is assigned a value we can  imagine that a box is created in the memory. This box has a sticker attached to it i.e. the variable name. Inside the box the value assigned to the variable is stored. Changes taking place in one does not affect the other.

      var x = 5;

      var y = x;

Here value is assigned to x and copied to y.  x and y aren’t connected and changes made to one doesn’t affect the other.

**Copy by reference:**

In case of a non-primitive data-type the values are not directly copied. When a non-primitive data-type is assigned a value a box is created with a sticker of the name of the data-type. However, the values it is assigned is not stored directly in the box. The language itself assigns a different memory location to store the data. The address of this memory location is stored in the box created.

So when the non-primitive variables are copied, the address of memory location is copied to the variable. So two variables becomes connected.

When we make changes to one variable, values are changed at the referenced memory location, resulting in the change in another variable also.

**3.How to copy by value a composite data type (array+objects).**

There are three ways to copy composite data types by value:

**1.** **Using spread  (…)** **operator:**

Spread operator allows iterable to expand in places where 0+ arguments are expected. Using this, we can expand variable array conveniently. Spread operator will clone your object, and then later we can assign that cloned object to another variable/object, thus achieving copy by value.

      var arr = [1, 2, 3];

      var copied\_arr = …arr;

**2.** **Using Object.assign():**

The **Object.assign()** method copies all enumerable own properties from one or more source objects to a target object. It returns the target object.

      var a = [1, 2, 3];

      var b = Object.assign([], a);

      console.log(b); //[1, 2, 3]

      b[2] = 4545;

      console.log(b); //[1, 2, 4545]

**3.**     **Using JSON.parse() and JSON.stringify():**

The JSON object, available in all modern browsers, has two useful methods to deal with JSON-formatted content: parse and stringify. JSON.parse() takes a JSON string and transforms it into a JavaScript object. JSON.stringify() takes a JavaScript object and transforms it into a JSON string.Using JSON.parse() and JSON.stringify() for copy performs deep copy .

**4.Print the total population of all countries in console**

Html file

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>api data</title>

</head>

<body>

    <script src ='script.js'></script>

</body>

</html>

Script.js

var xmlhttp = new XMLHttpRequest();

xmlhttp.open('GET','https://restcountries.eu/rest/v2/all', true);

xmlhttp.send();

xmlhttp.onload = function() {

    var arr = JSON.parse(this.response);

    for(var i in arr)

    {

     console.log(arr[i].population);

    }

}