Task 1 - Day 1

# 1. Difference between HTTP1.1 and HTTP2

# HTTP 1.1:

* For better understanding, let’s assume the situation when you make a request to the server for the google.html page & server responds to you as a resource google.html page.
* Before sending the request and the response there is a TCP connection established between client & server. again you make a request to the server for image img.jpg & the server gives a response as an image img.jpg.
* The connection was not lost here after the first request because we added a keep-alive header which is part of the request so there is an open connection between the server & client.
* There is a persistent connection which means several requests & responses are merged in a single connection. These are the drawbacks that lead to the creation of HTTP/2: The first problem is HTTP/1.1 transfer all the requests & responses in the plain text message form.
* The second one is head of line blocking in which TCP connection is blocked by all other requests until the response is not received. all the information related to the header file is repeated in every request.

HTTP 2:

* HTTP/2 was developed over the SPDY protocol. HTTP/2 works on the binary framing layer instead of textual that converts all the messages in binary format. it works on fully multiplexed, that is one TCP connection is used for multiple requests.
* HTTP/2 uses HPACK which is used to split data from header. it compresses the header. The server sends all the other files like CSS & JS without the request of the client using the PUSH frame.

| 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 requests behind it until it doesn’t get its all resources. | It allows multiplexing so one TCP connection is required for multiple requests. |
| It uses requests resource Inlining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| It compresses data by itself. | It uses HPACK for data compression. |

# 2. Objects in its internal Representation in JavaScript

* Objects, in JavaScript, is its 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 (depending on their types).
* Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.
* An object is a reference data type. Variables that are assigned a reference value are given a reference or a pointer to that value. That reference or pointer points to the location in memory where the object is stored. The variables don’t actually store the value.

**Internal Representation**:

* 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.
* 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.
* 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.
* 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.
* An object property name can be any valid JavaScript string, or anything that can be converted to a string, including the empty string. However, any property name that is not a valid JavaScript identifier can only be accessed using the square bracket notation. This notation is also very useful when property names are to be dynamically determined (when the property name is not determined until runtime).

# 3. Code Kata

* Done in Code kata platform