**Full Stack development Course – Day 1**

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

|  |  |
| --- | --- |
| **HTTP/1.1** | **HTTP/2** |
| Response and request are made in text format | Response and request are made in binary format |
| Requests are sent in a sequence over a single connection | Requests can be sent in parallel over a single connection. i.e. asynchronous requests can be made |
| There is possibility that when response is not received for request, it can block the other requests in the queue | Supports multiplexing which means that one request can get response even another request is waiting or does not get response |
| Header compression is not possible | Uses header compression to reduce the size of messages |
| There can be only one request outstanding at a time on a connection | There can be Multiple requests outstanding at the same time on a connection |
| works only on pull request and response. i.e. if the client requests and the server provide only the response the client asked. | Support push request and response. i.e. server sends response even if the user is not going use the response immediately. Push request are important in case of time sensitive data |
| Uses a single connection for each request and response cycle | Uses a single TCP connection for multiple requests and response |

**Task 2 – Objects and their internal representation in Java7Script**

Objects are one of the 8 JavaScript data types. They are the most important datatype in JS. Understanding objects is very important to write an effective code

Objects are internally represented in JS by their properties which are the key and value pairs.

Keys are in the form of string whereas values can be of any data types like numbers, strings, Boolean, regular expression, dates etc. An object itself can contain another object as value pair. Objects properties can be created, accessed, modified or removed which makes them mutable compared to other data types.

**Ways to Create JS Object**

* The simplest way is using the literal notation

e.g. Here we define the object and create key:value pairs in one statement

const customer = {

name:”ABC”,

DOB:”02-02-1995”,

gender:”male”

};

* Create an empty object and then add the key:value pairs.

const customer = {};

customer.name = “ABC”;

customer.DOB = “02-02-1995”;

customer.gender = “male”;

* Creating an object using new Object method

const customer = new Object();

customer.name = “ABC”;

customer.DOB = “02-02-1995”;

customer.gender = “male”;

* Creating an object using new constructor function

function Customer(name,DOB,gender){

this.name = name;

this.DOB = DOB;

this.gender = gender;

}

var cust1 = new Customer('ABC','02-02-1995','male');

**Accessing JS object values**

There are two ways to access Object values

* Using the dot notation objectname.property name

const customer = {

name:”ABC”,

DOB:”02-02-1995”,

gender:”male”

};

e.g. customer.name or customer.DOB or customer.gender

* Using bracket notation objectname[‘propertyname’]

customer[‘name’] or customer[‘DOB’] or customer[‘gender’]