DIFFERENCE BETWEEN HTTP1.1 vs HTTP2

HTTP-1.1

\* The client has to send a request to the server using the method (GET/POST).

\* Server responds with the requested resource, for example – image, alongside the status of what it did to the client’s request.

\* This is not a one-time process. Such requests and responses needs to be transferred between both these machines until the client receives all the resources, essential to load a web page on the end-user’s screen.

\* It doesn’t' have binary framing layer.

HTTP-2

\* HTTP/2 was released at Google as the significant improvement of its predecessor. It was initially

modeled after the SPDY protocol and went through significant changes to include features like

multiplexing, header compression and stream prioritization to minimize page load latency.

\* HTTP/2 uses a binary framing layer. This layer encapsulates messages and converted to its

binary equivalent while making sure that its HTTP semantics remain untamed. This feature

of HTTP/2 enables gRPC to use lesser resources.

***OBJECTS AND ITS INTERNAL REPRESENTATION IN JAVASCRIPT***

1. Object is defined as an unordered collection of related data, of primitive or reference types, in the

form of “key: value” pairs.

Its internal representation are as follows:

- OBJECT LITERAL

Object literal is a comma-separated list of name-value pairs wrapped in curly braces.

Object literals encapsulate data, enclosing it in a tidy package.

EX :

let car={id:1 , name:’abc’ , display:function() }

- OBJECT.CREATE()

The method creates a new object, using an existing object as the prototype of the newly created object.

using the object literal example as prototype-

EX:

var car2 = Object.create(car);

car.id=2;

car.name = ‘xyz’;

- OBJECT CONSTRUCTOR

Useful when we require to create multiple objects of similar type. In this case, a constructor (kind of blueprint)

is created and multiple objects can be initialized using the new keyword using the constructor as a wrapper

for the newly created objects.

EX:

construction function-

function Person(name, age, eye) {

this.Name = name;

this.age = age;

this.eyeColor = eye;

}

creating objects using constructor-

var p1= new Person(“John”, 50, “blue”);

var p2= new Person(“Sally”, 48, “green”);

- OBJECT.ASSIGN()

It is used to copy the values and properties from one or more source objects to a target object. It invokes

getters and setters since it uses both [[Get]] on the source and [[Set]] on the target.

EX

Input : var obj1 = { a: 10 };

var obj2 = { b: 20 };

var obj3 = { c: 30 };

var new\_obj = Object.assign(o1, o2, o3);

console.log(new\_obj);

Output : Object { a: 10, b: 20, c: 30 }

-OBJECT.ENTRIES

This method transforms a list of key-value pairs into an object.

EX:

const entries = new car([

[‘id’, 4],

[‘color’, ‘blue’]

]);

const car1= Object.fromEntries(entries);

console.log(car1);

output: Object { id: 4, color: ‘blue’}

-ACCESSING OBJECT VALUES:

We can access the object values by using dot (.) notation and bracket ([]) notation.

If we are using bracket ([]) notation to access object values we have to specify the key

as string within quotes (“”).

EX:

var car={name:"Renault",color:"red"};

console.log("Name :"+car.name);//Name :Renault

console.log("Name :"+car["name"]);//Name :Renault