**Q1: HPPT 1 VS HTTP 2**

|  |  |  |
| --- | --- | --- |
| Differentiator | HTTP/1 | HTTP/2 |
| Year | 1991 | 2015 |
| Key Features | For every TCP connection there is only one request and one response. | Uses multiplexing, where over a single TCP connection resources to be delivered are interleaved and arrive at the client almost at the same time. |
| Status Code | Can define 16 status codes; the error prompt is not specific enough. | Underlying semantics of HTTP such as headers, status codes remains the same. |
| Authentication Mechanism | Uses basic authentication scheme which is unsafe since username and passwords are transmitted in clear text or base64 encoded. | Security concerns from previous versions will continue to be seen in HTTP/2. However, it is better equipped to deal with them due to new TLS features like connection error of type Inadequate Security. |
| Caching | Provides support for caching via the If-Modified-Since header. | HTTP/2 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 | TCP starts slowly and with domain shading (resources can be downloaded simultaneously by using multiple domains), connection reuse and pipelining, there is an increased risk of network congestion. | HTTP/2 utilizes multiplexing and server push to effectively reduce the page load time by a greater margin along with being less sensitive to network delays |

Q2. write a blog on objects and its internal representation in javascript

**Objects and properties**

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.

Example:

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.

**CREATING OBJECTS IN JAVASCRIPT**

**Create JavaScript Object with Object Literal**

One of the easiest way to create a JavaScript object is object literal, simply define the property and values inside curly braces as shown below

let bike = {name: 'SuperSport', maker:'Ducati', engine:'937cc'};

**Create JavaScript Object with Constructor**

Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavor as shown below

function Vehicle(name, maker) {  
 this.name = name;  
 this.maker = maker;  
}  
let car1 = new Vehicle(’Fiesta’, 'Ford’);  
let car2 = new Vehicle(’Santa Fe’, 'Hyundai’)  
console.log(car1.name); //Output: Fiesta  
console.log(car2.name); //Output: Santa Fe

**Using the JavaScript Keyword new**

The following example also creates a new JavaScript object with four properties:

Example

var person = new Object();  
person.firstName = “John”;  
person.lastName = “Doe”;  
person.age = 50;  
person.eyeColor = “blue”;

**Using the Object.create method**

Objects can also be created using the Object.create method. This method can be very useful, because it allows us to choose the prototype object for the object we want to create, without having to define a constructor function.

// Animal properties and method encapsulation  
var Animal = {  
 type: 'Invertebrates', // Default value of properties  
 displayType: function() { // Method which will display type of Animal  
 console.log(this.type);  
 }  
};  
// Create new animal type called animal1   
var animal1 = Object.create(Animal);  
animal1.displayType(); // Output:Invertebrates  
// Create new animal type called Fishes  
var fish = Object.create(Animal);  
fish.type = 'Fishes';  
fish.displayType(); // Output:Fishes