|  |  |  |
| --- | --- | --- |
|  | **HTTP 1.1** | **HTTP 2** |
| 1 | The first version of HTTP was created in 1997. | In 2015, a new version of HTTP called HTTP/2 was created. |
| 2 | It works on the textual format. | It works on the binary protocol. |
| 3 | 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. |
| 4 | It uses requests resource Inlining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| 5 | It compresses data by itself. | It uses HPACK for data compression. |

**TASK 1**

**1.Difference between HTTP 1.1 vs HTTP 2:**

**2.Objects and its internal representation:**

**Objects:**

An object in JavaScript is a collection of key-value pairs. Each key-value pair is called as a property. A property can be a function, an array, an object itself or any primitive data type i.e., integer, string, etc. Functions in object are called as methods.

**Example:**

var human = {

firstName : “Sri”

lastName : “Yazhini”

age : 22

}

Here firstName, lastName, and age are properties of the same object i.e. human.firstName is the key and Sri is the value of the property.

**Internal representation:**

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**

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. For example, let’s create an object named **mycar** and give it properties named **make, model,** and **year** as follows:

**var myCar = new Object();**

**myCar.make = ‘Ford’;**

**myCar.model = ‘Mustang’;**

**myCar.year = 1990;**

Unassigned properties of an object are **undefined** (and not **null**).

**myCar.color; // undefined**

Properties of JavaScript objects can also be accessed or set using a bracket notation (for more details see [property accessors](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Property_Accessors)). Objects are sometimes called *associative arrays*, since each property is associated with a string value that can be used to access it. So, for example, you could access the properties of the **myCar** object as follows:

**myCar['make'] = 'Ford';**

**myCar['model'] = 'Mustang';**

**myCar['year'] = 1990;**