**1. Difference between HTTP/1.1 and HTTP/2**

HTTP stands for hypertext transfer protocol & it is used in client-server communication. By using HTTP user sends the request to the server & the server sends the response to the user. There are several stages of development of HTTP but we will focus mainly on HTTP/1.1 which was created in 1997 & the new one is HTTP/2 which was created in 2015.

**HTTP/1.1**

Officially standardized in 1999, HTTP 1.1 is the most recent version of HTTP. Just like its predecessor, this application protocol transfers data between web servers and clients. However, what makes it different is the incorporation of PUT, DELETE, and OPTIONS, and it includes features like HTTP pipelining and chunked transfer encoding. Due to all these features, HTTP 1.1 provides better performance and more flexibility than HTTP 1.

### **Benefits of HTTP 1.1**

* **Persistent Connections:**
* **Improved Caching:**
* **More Efficient Request and Response Handling:**

## **HTTP 2**

HTTP 2 is the second major version of the HTTP network protocol used for transmitting data over the Internet. It was developed to improve the performance and efficiency of web applications by reducing the amount of data sent over the wire. This reduces the number of round trips required to load a webpage and allows for a more efficient application of resources.

### **Benefits of HTTP 2**

* Low overhead in parsing data — a critical value proposition in HTTP/2 vs HTTP1.
* Less prone to errors.
* Lighter network footprint.
* Effective network resource utilization.
* Eliminating security concerns associated with the textual nature of HTTP1.x such as response splitting attacks.

## **Why HTTP 2 Replaced HTTP 1.1**

### Request Multiplexing

### Header Compression

## **HTTP 2.0 vs. 1.1**

### Predicting Resource Requests

### Buffer Overflow

### Multiplexing

### Binary Protocol

|  |  |  |
| --- | --- | --- |
|  | **HTTP 1.1** | **HTTP 2** |
| **Development** | It was developed in the year 1997 | It was developed in the year 2015. |
| **Compression** | It compresses data by itself. | It uses HPACK for data compression. |
| **Binary Protocol** | A text-based protocol uses plain text to encode and transmit data. | It works on the binary protocol as a series of binary codes encode and transmit data rather than plain text. |
| **Security** | The client sends a request to a server, and the server sends a response back to the client. | A different underlying Remote Protocol 2 (SRP2) establishes a secure connection between a client and a server. |
| **Multiplexing** | A separate connection is established for each request and response, which can add overhead and latency to the communication process | It allows multiplexing so multiple requests and responses can be sent over a single connection. |
| **Buffer Overflow** | HTTP 1.1 cannot handle buffer overflow vulnerabilities due to the lack of sufficient measures. | HTTP 2 includes measures to prevent buffer overflow vulnerabilities. |
| **Performance** | HTTP 1.1 does not include any in-built features, so the performance it delivers is less efficient. | HTTP 2 is designed to be more efficient and performant than HTTP 1.1. This is because HTTP 2 includes several features like multiplexing, binary protocol and header compression. |

**Objects And Its Internal Representation In JavaScript**

Objects, in JavaScript, is it’s 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.

Loosely speaking, 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.

# ****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**

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:

Example

var myCar = new Object();

myCar.make = 'Ford';

myCar.model = 'Mustang';

myCar.year = 1969;

# Create JavaScript Object with Object Literal

# One of 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

Example

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”;