**HTTP 1.1 VS 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**

HTTP 1.1 was introduced in 1997, which allowed multiple requests and responses to be sent over a single TCP connection. However, HTTP 1.1 had its limitations:

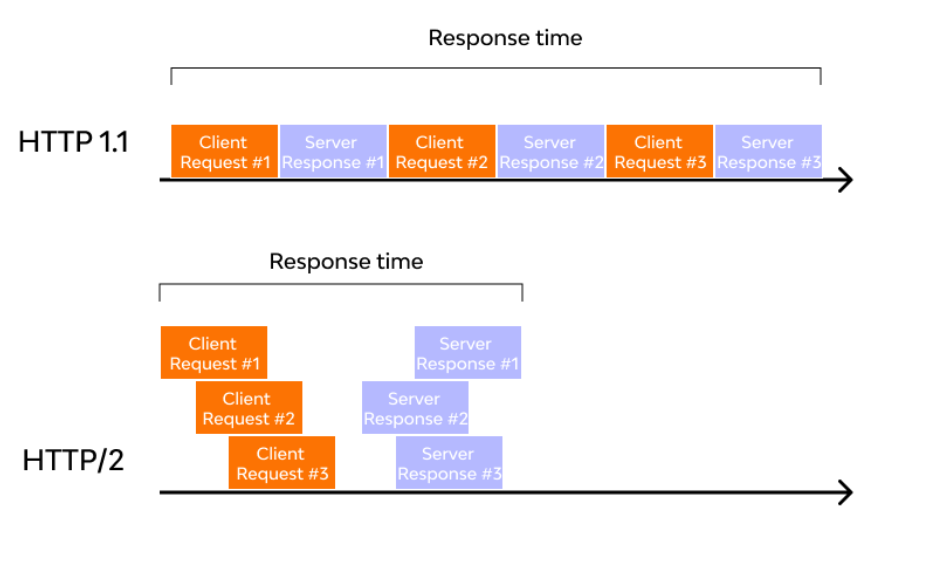
1. **Head-of-Line Blocking**: In HTTP 1.1, requests were served sequentially, leading to a phenomenon known as head-of-line blocking. If one resource took longer to fetch, it blocked subsequent resources from being downloaded, resulting in performance bottlenecks.
2. **Compression and Multiplexing**: While HTTP 1.1 supported gzip compression for reducing the size of transmitted data, it lacked native support for request and response multiplexing. Each request had to wait for the previous one to complete before it could be proceed

**http 2**

In response to the shortcomings of HTTP 1.1, HTTP 2.0 was standardized in 2015, bringing several enhancements to the table:

1. **Multiplexing**: HTTP 2.0 introduced multiplexing, allowing multiple requests and responses to be interleaved within a single TCP connection. This effectively eliminates head-of-line blocking and improves overall efficiency by utilizing available network resources more effectively.
2. **Binary Protocol**: Unlike HTTP 1.1, which transmitted data as text, HTTP 2.0 uses a binary framing layer to serialize requests and responses. This reduces overhead and improves parsing efficiency.
3. **Header Compression**: HTTP 2.0 employs header compression using the HPACK algorithm, reducing redundant header information and further optimizing data transmission.

| **HTTP/1.1** | **HTTP/2** |
| --- | --- |
| It works on the textual format. | It works on the binary protocol. |
| 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. |
| It uses requests resource Inlining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| It compresses data by itself. | It uses HPACK for data compression. |



**Conclusion:**

HTTP 2.0 represents a significant advancement over its predecessor, addressing many of the performance bottlenecks associated with HTTP 1.1. By introducing multiplexing, binary framing, and header compression, HTTP 2.0 optimizes data transmission and enhances the browsing experience for users. As the internet continues to evolve, protocols like HTTP will likely undergo further refinements to meet the demands of an increasingly interconnected world.

**OBJECTS AND ITS INTERNAL REPRESENTATIONS**

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).

 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.

For Eg. If your object is a student, it will have properties like name, age, address, id, etc

# **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:

Creating objects

1. Object constructor

Instantiate an object using the Object() constructor

JavaScript variables, both the object name and property name are case sensitive. You can define a property by assigning it a value

Example:

var myCar = new Object();

myCar.make = 'Ford';

myCar.model = 'Mustang';

myCar.year = 1969;

2.Object literal syntax:

Create an object directly using {}

Example:

var myCar = {

make: 'Ford',

model: 'Mustang',

year: 1969

};

## Internal Representation of Objects

objects are stored as a collection of key-value pairs, where the keys are the property names and the values can be data or references to functions.

When you access an object’s property or method, JavaScript searches for the property name in the object’s internal collection of key-value pairs. If it finds a match, it returns the corresponding value.