Task 1:

1. Write a blog on Difference between HTTP1.1 vs HTTP2
2. Write a blog about objects and its internal representation in Javascript
3. codekata practice
4. Read about IP address, port, HTTP methods, MAC address

<https://github.com/reach2arunprakash/javascript-101/tree/master/ppt>

1. **Write a blog on Difference between HTTP1.1 vs HTTP2**

Difference between HTTP1.1 vs HTTP2

|  |  |  |
| --- | --- | --- |
|  | HTTP1.1 | HTTP2 |
| 1. | Text format is used | Binary format is used.  In HTTP/2, the binary framing layer encodes requests/responses and cuts them up into smaller packets of information, greatly increasing the flexibility of data transfer. |
| 2. | Other Requests are blocked till the first request’s response is completed.  For example – If a request at the queue head cannot retrieve its required resources, it can block all requests behind it. This phenomenon is called head-of-line blocking (HOL blocking). | All the requests and responses are handled by one TCP connection. So requests are not blocked. |
| 3. | It compresses data by itself. | It uses HPACK for data compression. |

**Development Journey: Advancement from HTTP/1 to HTTP/2**

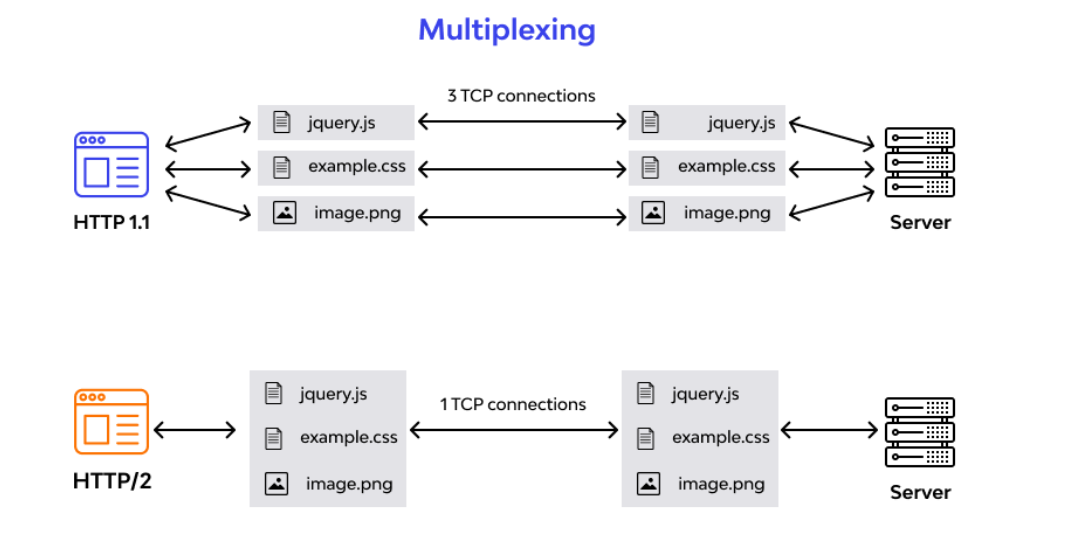
In 1989, Tim Berners-Lee invented HTTP. HTTP/1.1 was its 1st standardized version that was available for use in the year 1997 for the end-users. This version presented considerable performance optimization over its precursors and changed how communication was handled between clients and servers. However, its key qualities opened the doors to many performance and API security loopholes.

HTTP/1 was known to have poor response time. With websites becoming more resource-intensive, the protocol was losing its efficiency. It progressively became essential to minimize latency and boost page load speeds.

Google looked into these problems. And as expected, SPDY - an experimental project to end troubles with HTTP/1.x – was put into trial in the year 2010.

Years later, the IETF, Google, Microsoft, and Facebook released the fully-comprehensive and well-tested newer version of HTTP in 2015.

HTTP/2, based on SPDY protocol, was developed to address the inherent limitations of HTTP/1.1 and further progress the Internet.



## What is the HTTP/2 Protocol

HTTP/2 is the second version of HTTP with most of the shortcomings of its predecessor addressed in it. It has come with advancements in efficiency, speed, and security. Till the date, HTTP/2 is supported on almost all popular web browsers, such as Chrome, Firefox, Internet Explorer, and Safari.

HTTP/2 aims at simplifying, speeding up, and empowering the applications across the internet. To achieve the same, the protocol emphasizes on page load time, resource optimization, and round-trip time (RTT) reduction.

For resource-heavy pages, it supports gradual downloading on the user’s end to improve user experience.‍

## HTTP/1.1 and HTTP/2 Main Differences

Launching of the HTTP/2 was an attempt to overcome the limitations of HTTP/1.1 and make it a more efficient web protocol. So, the major differences in these two are mainly the additions or upgrades applied in HTTP/2. Let’s see what they are:

### The Background

For better contextualization of the certain alterations that HTTP/2 made to its precursor, we’ll take a quick look at their basic functionalities and development details first.

HTTP/1.1

HTTP protocol was developed in 1989 as the common language that enables client and server machines’ interaction. Process steps are as enlisted:

1. The client (browser) has to send a request to the server using the method (GET/POST).
2. Server responds with the requested resource, for example – image, alongside the status of what it did to the client’s request.

Keep in mind that 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 (your) screen.

This request-response exchange can be regarded as an IP stack being handled by transfer layer and networking layers before finally reaching to the application layer. Now, let’s see how HTTP/2 handles the same scenario.

**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. After its release, Google announced that it would not provide support for SPDY in favor of HTTP/2.

The major feature that differentiates HTTP/2 from HTTP/1.1 is the binary framing layer. Unlike HTTP/1.1, HTTP/2 uses a binary framing layer. This layer encapsulates messages – converted to its binary equivalent – while making sure that its HTTP semantics (method details, header information, etc.) remain untamed. This feature of HTTP/2 enables [gRPC](https://www.wallarm.com/what/the-concept-of-grpc) to use lesser resources.

### Delivery Models

As discussed before, HTTP/1.1 sends messages as plain text, and HTTP/2 encodes them into binary data and arranges them carefully. This implies that HTTP/2 can have various delivery models.

Most of the time, a client's initial response in return for an HTTP GET request is not the fully-loaded page. Fetching additional resources from the server requires that the client send repeated requests, break or form the TCP connection repeatedly for them.

As you can conclude already, this process will consume lots of resources and time.

**HTTP/1.1**

HTTP/1.1 addresses this problem by creating a persistent connection between server and client. Until explicitly closed, this connection will remain open. So, the client can use one TCP connection throughout the communication sans interrupting it again and again.

This approach surely ensures good performance, but it also is problematic.

For example – If a request at the queue head cannot retrieve its required resources, it can block all requests behind it. This phenomenon is called head-of-line blocking (HOL blocking).

From the above, we can conclude that multiple TCP connections are essential.

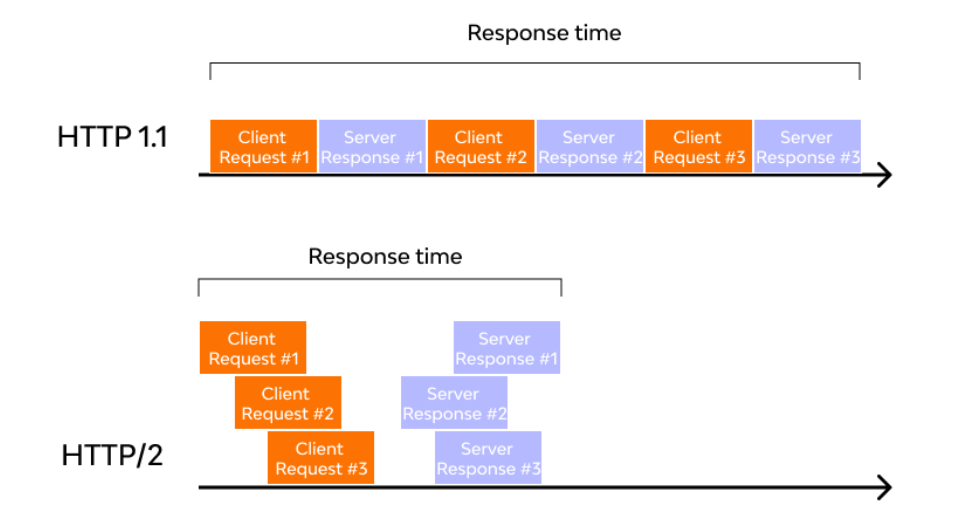
**HTTP/2**

Considering the bottleneck in the previous scenario, the HTTP/2 developers introduced a binary framing layer. This layer partitions requests and responses in tiny data packets and encodes them. Due to this, multiple requests and responses become able to run parallelly with HTTP/2 and chances of HOL blocking are bleak.

Not only has it solved the HOL blocking problem in HTTP/1.1, but it also concurrent message exchange between the client and the server. This way, both of them can have more control while the connection management quality is boosted too.

The problems of HTTP/1.1 looks resolved to a great extent here. However, at times, multiple data streams demanding the same resource can hinder HTTP/2’s performance. To achieve better performance, HTTP/2 has another way. It has the capability of stream prioritization.

When sending streams in parallel, the client can assign weights (1-256) to its stream to prioritize the responses it demands. Here, the higher the weight, the higher the priority. The serve sets the data retrieval order as per the request’s weight. Programmers can enjoy better control on page rendering process with stream prioritization ability.



1. **Write a blog about objects and its internal representation in Javascript**

Objects are important data types in javascript. Objects are different than primitive datatypes (i.e. number, string, boolean, etc.). Primitive data types contain one value but Objects can hold many values with different data types in form of Key: value pair. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object. An object can contain another Object and even an array.

In array values are accessed based on the index. It is very tough to remember what array[0] has.

But in Object accessing the value is in human readable format object[‘name’] with key-value pair.

const myArray =[

‘prem’,

32

‘Cbe’

]

console.log(myArray[0]);

const myObject = {

Name : ‘prem,

Age : ‘32’

Location : ‘Trichy’

}

console.log(myObject [‘Name’]);

console.log(myObject.Name);

Object values can be accessed through -> 1)Dot notation 2)Bracket Notation

**Creating Object using new Keyword.**

var myBike = new Object();  
myBike.make = 'Bajaj';  
myBike.model = ‘2021’;

Unassigned properties of an object are [undefined](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined) (and not [null](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/null)).

myBike.color; // undefined

**Introduction to Objects**

// In an array only index is available and there is no Key-Value pair

// To solve this Object data structure came into the picture with Key-Value pair

*const* yourOwnDetailsArray = ['Mathan', 21, 'Playing Cricket', 'Student'];

*const* myOwnFatherDetailsObject = { fatherName: 'Rajan', fatherAge: 77, fatherHobby: 'Listening Music', fatherJob: 'Teacher' };

*const* myOwnName = 'Prem';

*const* myOwnDetails = [

    myOwnName,

    2024 - 1974,

    'Playing Shuttle',

    'Tech Lead',

    yourOwnDetailsArray

];

console.log(myOwnDetails);

console.log(myOwnDetails[4][0]);

console.log(myOwnDetails[4][1]);

console.log(myOwnDetails[4][2]);

*const* myOwnDetailsObject = {

    myName: myOwnName,

    myAge: 2024 - 1974,

    myHobby: 'Playing Shuttle',

    myJob: 'Tech Lead',

    myFriend: yourOwnDetailsArray,

    myFather: myOwnFatherDetailsObject

};

console.log('From Object data type myName: ' + myOwnDetailsObject.myName);

console.log('From Object data type friend name: ' + myOwnDetailsObject.myFriend[0]);

console.log(`From Object data type father's name: ` + myOwnDetailsObject.myFather.fatherName);

//In array values of elements placed in the indexed order

//But in Object there is no need of order.Just we can access through the human //readable Key to obtain the value

//Object : Access data through dot and bracket Notation

*const* myPersonalDetailsObject = {

    Name: myOwnName,

    Age: 2024 - 1974,

    Hobby: 'Playing Shuttle',

    Job: 'Tech Lead',

    Friend: yourOwnDetailsArray,

    Father: myOwnFatherDetailsObject

};

//Automatically ordered based on the Alphabetical

console.log(myPersonalDetailsObject);

console.log('Dot :' + myPersonalDetailsObject.Name);

console.log('Bracket Notation : ' + myPersonalDetailsObject['Name']);

//What is the use of Bracket Notation

*const* objectKeyName = prompt('Enter any key of myPersonalDetailsObject and we will get the value')

console.log(myPersonalDetailsObject[objectKeyName]);

if (myPersonalDetailsObject[objectKeyName]) {

    console.log(myPersonalDetailsObject[objectKeyName]);

}

else {

    console.log('Key is not available in the Object');

}