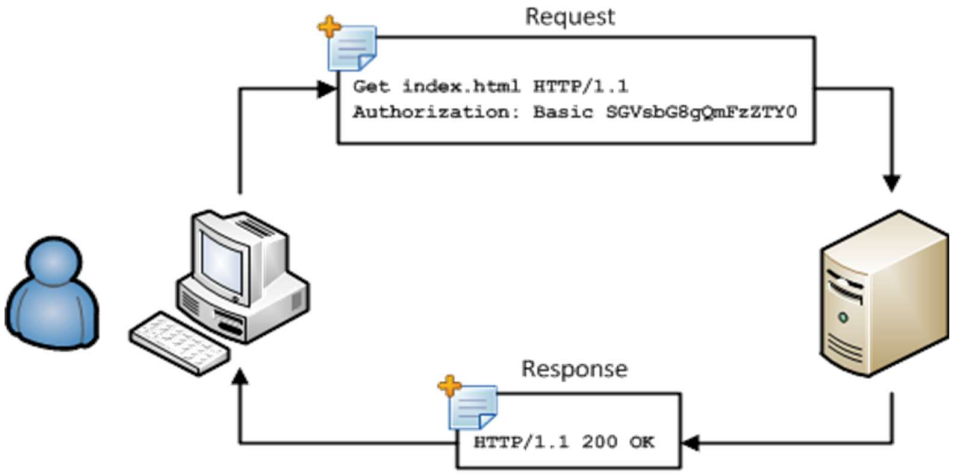
1.Write a blog on Difference between HTTP1.1 vs HTTP2

INTRODUCTION OF HTTP/1.1 and HTTP/2:

In 1989, Tim Berners-Lee invented HTTP. The first usable version of HTTP was created in 1997. Because it went through several stages of development, this first version of HTTP was called HTTP/1.1. This version is still in use on the web. In 2015, a new version of HTTP called HTTP/2 was created.

HTTP stands for hypertext transfer protocol, and it is the basis for almost all web applications. More specifically, HTTP is the method computers and servers use to request and send information. For instance, when someone navigates to cloudflare.com on their laptop, their web browser sends an HTTP request to the Cloudflare servers for the content that appears on the page. Then, Cloudflare servers send HTTP responses with the text, images, and formatting that the browser displays to the user.

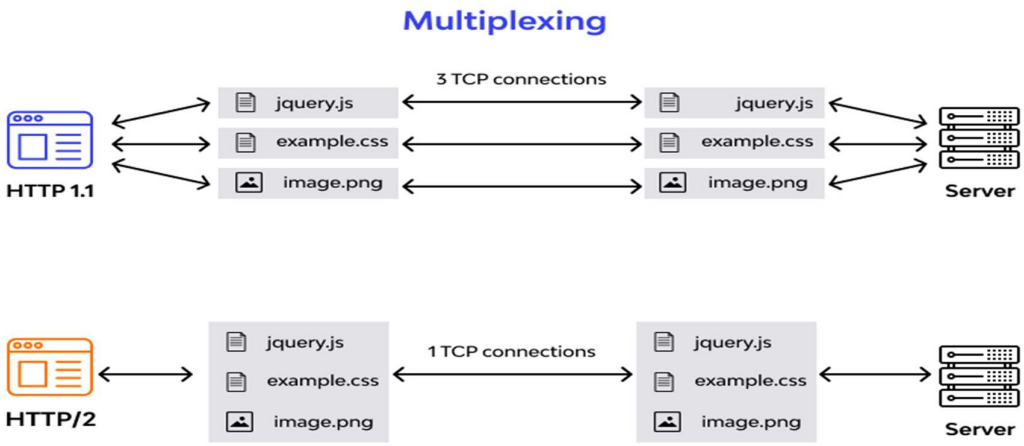


HTTP is a set of rules that runs on top of the TCP/IP suite of protocols and defines how files are to be transferred between clients and servers on the world wide web. HTTP/2 solves several problems that the creators of HTTP/1.1 did not anticipate. In particular, HTTP/2 is much faster and more efficient than HTTP/1.1. One of the ways in which HTTP/2 is faster is in how it prioritizes content during the loading process.

HTTP/1.1 and HTTP/2 Differences:

➢ Multiplexing:

HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. In contrast, HTTP/2 is able to use a single TCP connection to send multiple streams of data at once so that no one resource blocks any other resource. HTTP/2 does this by splitting data into binary-code messages and numbering these messages so that the client knows which stream each binary message belongs to.



➢ Server push:

Typically, a server only serves content to a client device if the client asks for it. However, this approach is not always practical for modern webpages, which often involve several dozen separate resources that the client must request. HTTP/2 solves this problem by allowing a server to "push" content to a client before the client asks for it. t introduces the concept of a server push where the server anticipates the resources that will be required by the client and pushes them prior to the client making requests. The client retains the authority to deny the server push; however, in most cases, this feature adds a lot of efficiency to the process.

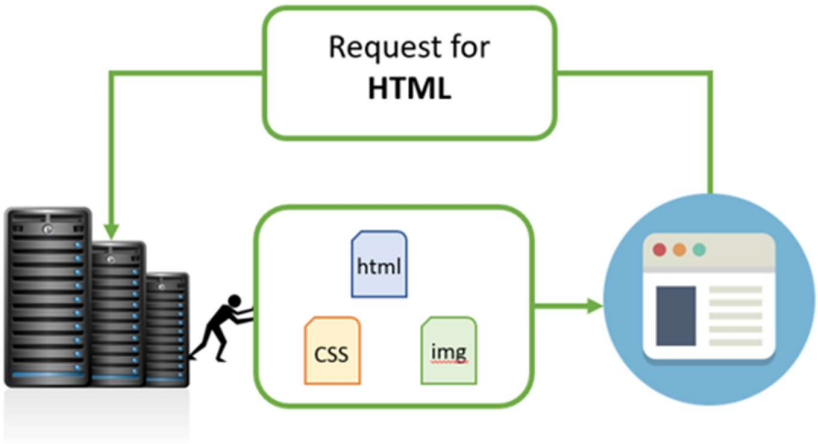


Figure 3: Server Push in HTTP/2

- **Header compression:**

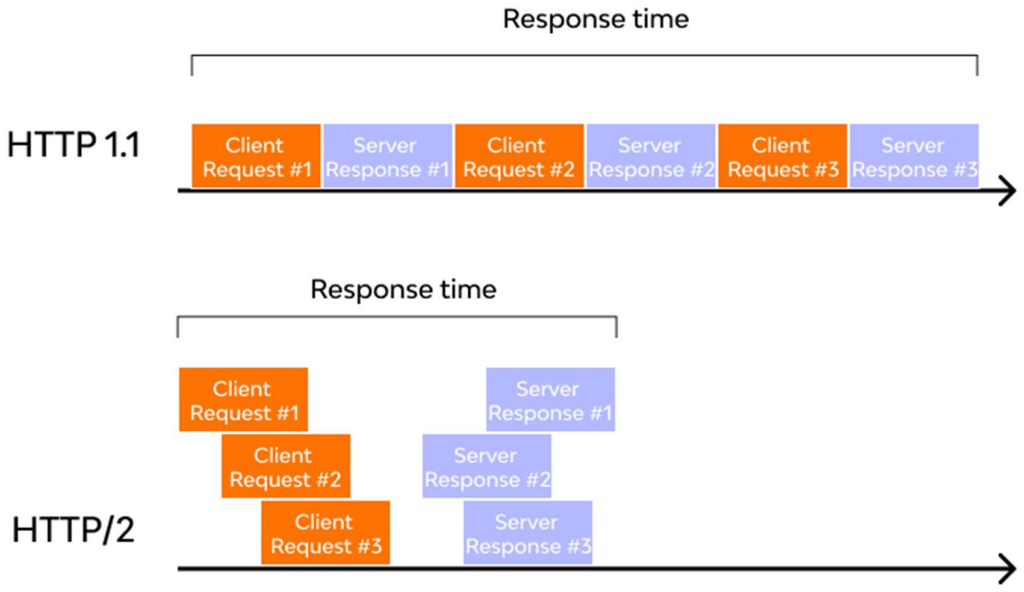
Small files load more quickly than large ones. To speed up web performance, both HTTP/1.1 and HTTP/2 compress HTTP messages to make them smaller. However, HTTP/2 uses a more advanced compression method called HPACK that eliminates redundant information in HTTP header packets. This eliminates a few bytes from every HTTP packet. Given the volume of HTTP packets involved in loading even a single webpage, those bytes add up quickly, resulting in faster loading.

- **Background:**

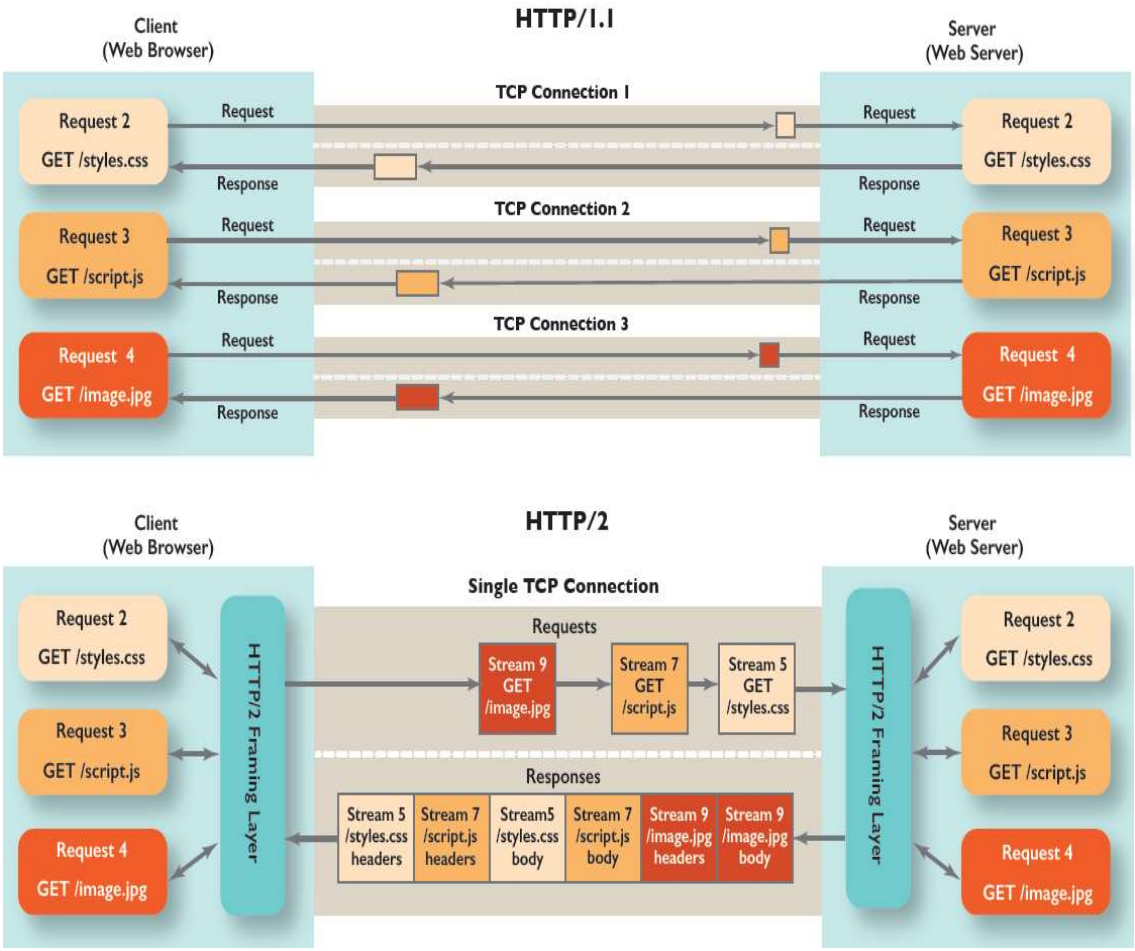
The major feature that differentiates HTTP/2 from HTTP/1.1 is the binary framing layer. Unlike HTTP/1.1, HTTP/2 uses a binary framing layer. This layer encapsulates messages – converted to its binary equivalent – while making sure that its HTTP semantics (method details, header information, etc.) remain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

➢ Response Time:

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.



Flow Chart:


HTTP/1.1
Client
(Web Browser)
Server
(Web Server)
TCP Connection 1
TCP Connection 2
TCP Connection 3
Request 2
GET /styles.css
Request 3
GET /script.js
Request 4
GET /image.jpg
Request
Response
HTTP/2
Single TCP Connection
Requests
Responses
Stream 9 GET /image.jpg
Stream 7 GET /script.js
Stream 5 GET /styles.css
Stream 5 /styles.css headers
Stream 7 /script.js headers
Stream5 /styles.css body
Stream 7 /script.js body
Stream 9 /image.jpg headers
Stream 9 /image.jpg body
HTTP/2 Framing Layer

2. Write a blog about 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.A JavaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

'JavaScript is an object-based language. Everything is an object in JavaScript'

➢ Creating Objects in JavaScript:

There are 3 ways to create objects.

- By object literal
- By creating instance of Object directly (using new keyword)
- By using an object constructor (using new keyword)

✓ JavaScript Array:

JavaScript array is an object that represents a collection of similar type of elements.The syntax of creating array using array literal is given below:

```
var arrayname=[value1,value2.....valueN];
```

As you can see, values are contained inside [ ] and separated by , (comma).

Example :

```
<script>
   var emp=["Sonoo","Vimal","Ratan"];
   for (i=0;i<emp.length;i++){
   document.write(emp[i] + "<br/>");
   }
   </script>
```

✓ JavaScript String:

The JavaScript string is an object that represents a sequence of characters. The string literal is created using double quotes.

```
var stringname="string value";
```

Example:

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

✓ Javascript Boolean:

JavaScript Boolean is an object that represents value in two states: true or false. You can create the JavaScript Boolean object by Boolean() constructor as given below.

Boolean b=new Boolean(value);

The default value of JavaScript Boolean object is false.

Example:

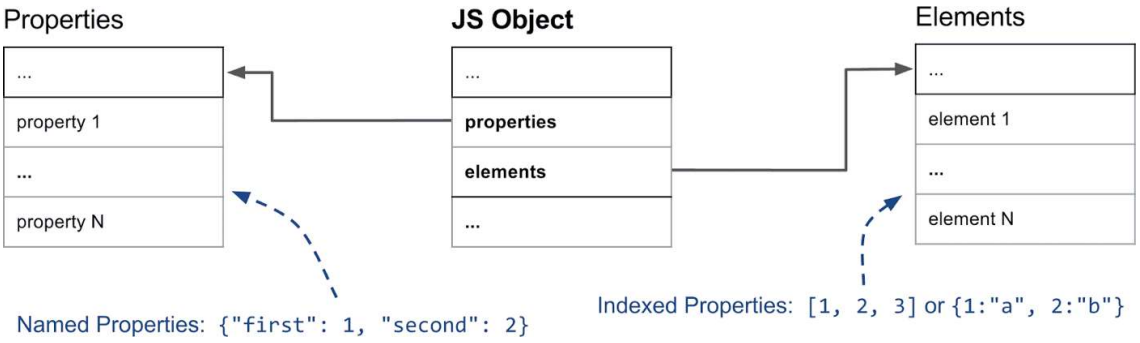
```
<script>
document.write(10<20);//true
document.write(10<5);//false
</script>
```

➢ JavaScript Objects and its internal representation

Unlike objects in other programming languages JavaScript has an unique approach towards objects and their representation. Objects in js are a collection of key-value pair.The collection can also include functions, which are called member functions and all the other keys are called its properties. Keys can either be a number or string,

```
var obj = {
name: “sam”,
age: 20,
list: [1,2,3],
changeAge: ()=>{
//logic to change age }
}
```

Here ‘name’ , ‘age’ are the properties and and ‘changeAge’ is called the member function of the object. The key-value pairs can also include array (i.e list). This is why objects in js are called an unordered collection of key-value pairs of primitive or reference types.



Accessing Object elements:

Keys are used to access all the elements in an object. They can be accessed in two ways:

* Dot notation

* Array notation (i.e []).

An object can be created with figure brackets {…} with an optional list of properties. A property is a “key: value” pair, where a key is a string (also called a “property name”), and value can be anything. Let us visualize this with the following syntax for creating an object in JavaScript.