**2. Objects and its internal representation in Javascript**

Object In JavaScript, an object is a standalone entity, with properties and type. Compare it with a cup, for example. A cup is an object, with properties. A cup has a color, a design, weight, a material it is made of, etc. The same way, JavaScript objects can have properties, which define their characteristics.

There are many ways to create objects in javascript as below

1. **Object constructor:**

The simplest way to create an empty object is using the Object constructor. Currently this approach is not recommended.

var object = new Object();

1. **Object's create method:**

The create method of Object creates a new object by passing the prototype object as a parameter

var object = Object.create(null);

1. **Object literal syntax:**

The object literal syntax is equivalent to create method when it passes null as parameter

var object = {};

1. **Function constructor:**

Create any function and apply the new operator to create object instances,

function Person(name){

var object = {};

object.name=name;

object.age=21;

return object;

}

var object = new Person("Sudheer");

1. **Function constructor with prototype:**

This is similar to function constructor but it uses prototype for their properties and methods,

function Person(){}

Person.prototype.name = "Sudheer";

var object = new Person();

This is equivalent to an instance created with an object create method with a function prototype and then call that function with an instance and parameters as arguments.

function func {};

new func(x, y, z);

**(OR)**

// Create a new instance using function prototype.

var newInstance = Object.create(func.prototype)

// Call the function

var result = func.call(newInstance, x, y, z),

// If the result is a non-null object then use it otherwise just use the new instance.

console.log(result && typeof result === 'object' ? result : newInstance);

1. **ES6 Class syntax:**

ES6 introduces class feature to create the objects

class Person {

constructor(name) {

this.name = name;

}

}

var object = new Person("Sudheer");

1. **Singleton pattern:**

A Singleton is an object which can only be instantiated one time. Repeated calls to its constructor return the same instance and this way one can ensure that they don't accidentally create multiple instances.

var object = new function(){

this.name = "Sudheer";

}

1. Difference between HTTP1.1 vs HTTP2

Hypertext Transfer Protocol (HTTP) is an application protocol that is, currently, the foundation of data communication for the World Wide Web.

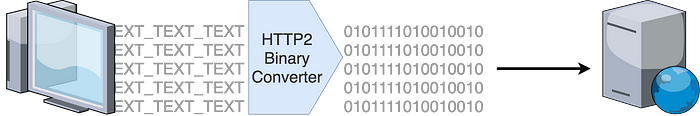
The first usable version of Introduction of HTTP was created in 1997. Because it went through several stages of development, this first version of HTTP was called HTTP/1.1. This version is still in use on the web. In 2015, a new version of HTTP called HTTP2 was created.

**High level Differences between HTTP/2 and HTTP/1.1:**

* HTTP2 is binary, instead of textual
* HTTP2 is fully multiplexed, instead of ordered and blocking
* HTP2 uses header compression to reduce overhead
* HTTP2 allows servers to “push” responses proactively into client caches

**Binary Protocol :**

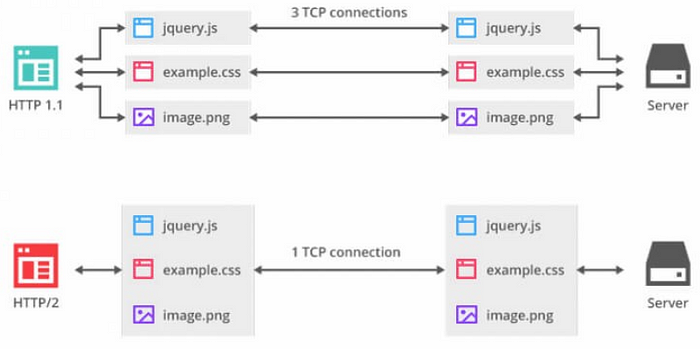
HTTP1.1 used to process text commands to complete request-response cycles. HTTP/2 will use binary commands (in 1s and 0s) to execute the same tasks. This attribute eases complications with framing and simplifies implementation of commands that were confusingly intermixed due to commands containing text and optional spaces.



HTTP2 Binary Protocol

**Multiplexing :**

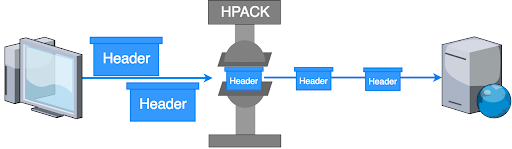
HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. In contrast, HTTP/2 is able to use a single TCP connection to send multiple streams of data at once so that no one resource blocks any other resource. HTTP/2 does this by splitting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



HTTP2 Multiplexing

**Header compression:**

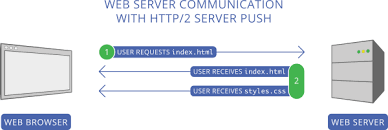
Small files load more quickly than large ones. To speed up web performance, both HTTP/1.1 and HTTP/2 compress HTTP messages to make them smaller. However, HTTP/2 uses a more advanced compression method called HPACK that eliminates redundant information in HTTP header packets. This eliminates a few bytes 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.



HTTP2 Hpack compression

**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. Server Push capability allows the server to send additional cacheable information to the client that isn’t requested but is anticipated in future requests. For example, if the client requests for the resource X and it is understood that the resource Y is referenced with the requested file, the server can choose to push Y along with X instead of waiting for an appropriate client request.



HTTP2 server push