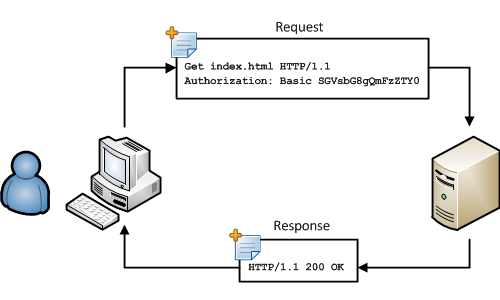
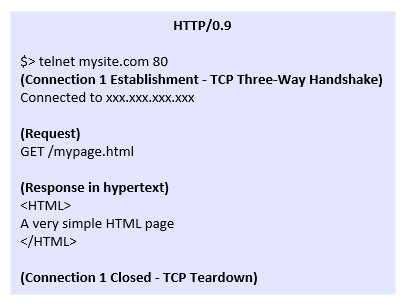
**Difference between HTTP1.1 vs HTTP2**

HTTP (Hypertext Transfer Protocol) 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.



The Beginning of HTTP: Version 0.9 & 1.0

In the earliest phase (HTTP/0.9), the HTTP protocol did not use headers and only transmitted plain HTML files. It was a one-line protocol only supporting the GET method.



**HTTP1.1**

HTTP/1.1, the first standardized version of HTTP, was introduced in 1997. It presented significant performance optimizations (over HTTP/0.9 and HTTP/1.0) and transformed the way requests and responses were exchanged between clients and servers.

It supports connection reuse i.e. for every TCP connection there could be multiple requests and responses, and pipelining where the client can request several resources from the server at once. However, pipelining was hard to implement due to issues such as head-of-line blocking and was not a feasible solution.

Introduces a warning header field to carry additional information about the status of a message. Can define 24 status codes, error reporting is quicker and more efficient.

It is relatively secure since it uses digest authentication, NTLM authentication.

Expands on the caching support by using additional headers like cache-control, conditional headers like If-Match and by using entity tags.

**HTTP2**

It 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.

Uses multiplexing, where over a single TCP connection resources to be delivered are interleaved and arrive at the client almost at the same time. It is done using streams which can be prioritized, can have dependencies and individual flow control. It also provides a feature called server push that allows the server to send data that the client will need but has not yet requested.

Underlying semantics of HTTP such as headers, status codes remains the same.

Security concerns from previous versions will continue to be seen in HTTP/2. However, it is better equipped to deal with them due to new TLS features like connection error of type Inadequate\_Security.

HTTP/2 does not change much in terms of caching. With the server push feature if the client finds the resources are already present in the cache, it can cancel the pushed stream.

HTTP/2 utilizes multiplexing and server push to effectively reduce the page load time by a greater margin along with being less sensitive to network delays.

**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

Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.

If your object is a student, it will have properties like name, age, address, id, etc and methods like updateAddress, updateNam, etc.

# **Objects and properties**

A property of an object can be explained as a variable that is attached to the object.

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

var myCar = new Object();  
myCar.make = 'Ford';  
myCar.model = 'Mustang';  
myCar.year = 1969;

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)).

myCar.color; // undefined

# **Creating Objects In JavaScript**

let bike = {name: 'SuperSport', maker:'Ducati', engine:'937cc'};

# **Create JavaScript Object with Constructor**

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