# 1.Difference between HTTP1.1 vs HTTP2

* HTTP/2 uses weighted prioritization to maximize perceived and actual page load speed to a degree that was not possible in HTTP/1.1.
* HTTP/1.1 transfers all the requests & responses in the plain text message form, Whereas HTTP/2 works by splitting data into binary-code messages.
* HTTP/1.1 loads resources one after the other, Whereas HTTP/2 uses multiplexing to send multiple streams of data at once. Hence no resource blocking in HTTP/2.
* HTTP/1.1 uses request resource Inlining for getting multiple pages, whereas HTTP/2 uses PUSH frame by server that collects multiple pages without repeated requests .
* HTTP/2 uses HPACK compression that eliminates redundant information in HTTP header packets. Hence resulting in faster loading than HTTP/1.1 which compresses data by itself.

# 2.Objects and its internal representation in Javascript

## Object:

* Objects in 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.
* Objects are sometimes called *associative arrays*, since each property is associated with a string value that can be used to access it.

## Creating Objects in JavaScript:

* By object literal
* By creating instance of Object directly (using new keyword)

## Using an Object Literal

* Using an object literal, we can both define and create an object in one statement.
* An object literal is a list of name:value pairs inside curly braces {}.

Example: const person = { firstName: "John",

lastName: "Doe",

age: 50,

eyeColor: "blue"};

## Using the JavaScript Keyword new

* We can create a new JavaScript object using new Object()

Example: const person = new Object();

person.firstName = "John";

person.lastName = "Doe";

person.age = 50;

person.eyeColor = "blue";

## JavaScript Objects are Mutable

* Objects are mutable: They are addressed by reference, not by value.

Example: const x = person; // Will not create a copy of person.

* The object x is not a copy of person.Both x and person are the same object.changes to x will also change person, because x and person are the same object.

