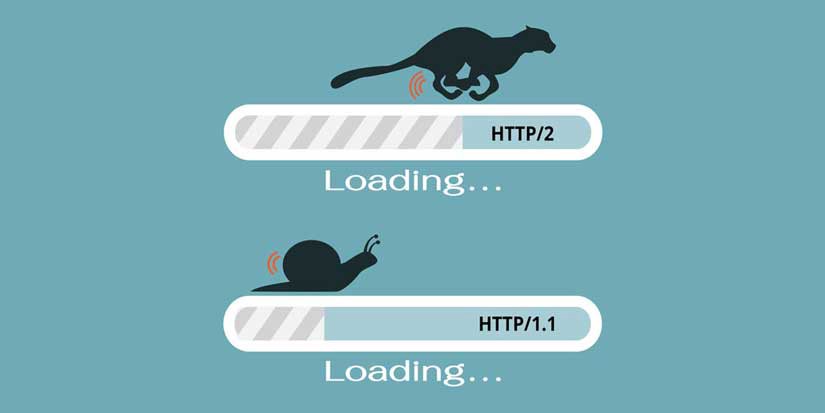
**1.HTTP 1.0 V/S HTTP/2**

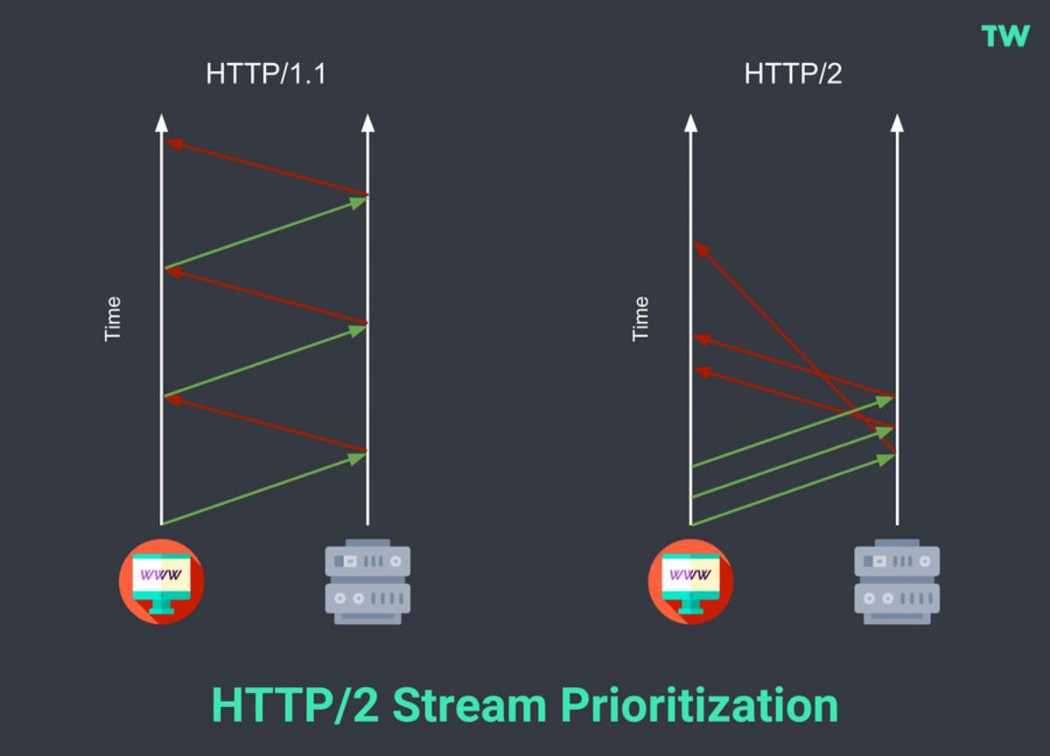
Digging up into history we know that the first version of HTTP was released in 1991 known as v0.9, although this supported only get method, later on in 1996 HTTP/1.0 was released. This was more efficient and secure due to additional methods, header fields, and also tied up with a security protocol. Now some more standards were added to it and by June 1996 which were adopted by 65% of the browsers at that time and this was officially released as HTTP/1.1 in 1997, so most of the browsers were using HTTP/1.1 even before its official release. Later on, many updates and standards were added to HTTP/1.1 until 2014 finally, a new more efficient version HTTP/2.0 was released in 2015.

Now, let’s jump to the differences between both the versions and why 98% of the browsers and 48% of the top 10 million websites support HTTP/2.0 within 4 years of release.

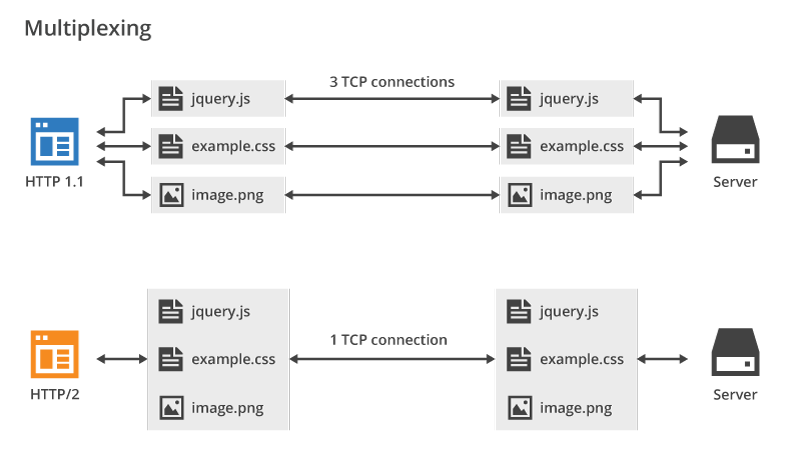


Now, you might have heard HTTP/2.0 is way much **faster** and **efficient** delivering much **better performance** compared to HTTP/1.1, yes it is can be verified [here](http://www.http2demo.io/), but the question is **how?.**

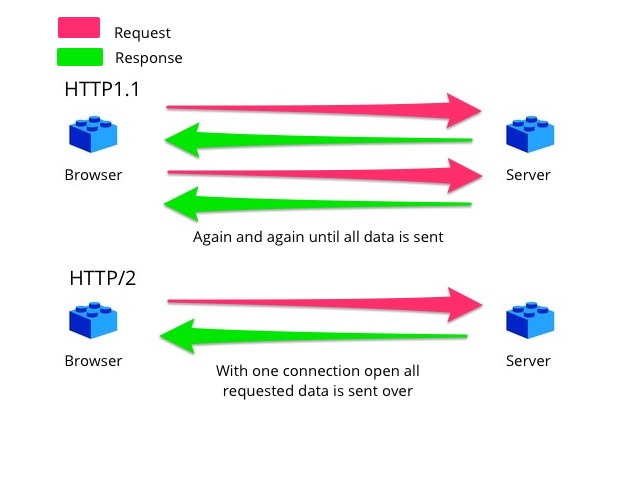
**Prioritization**- HTTP/2.0 uses prioritization i.e it prioritize the content while loading so, the most important content to the user is loaded first which makes the user perceive that the page is loading faster, even the developer has the authority to give weighted priority and decide in what order the content on the page should get loaded.



**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](https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/) connection to send multiple streams of data at once so that no one resource blocks any other resource.

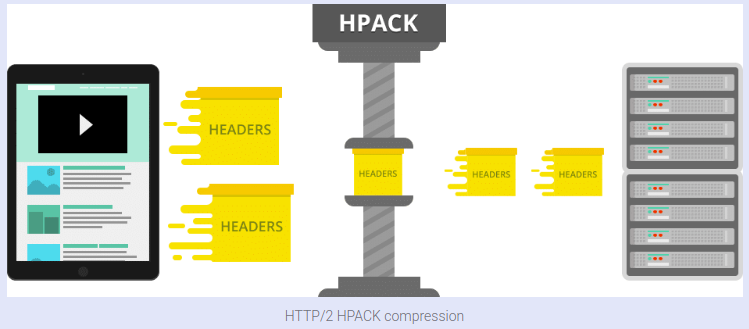


**Server push**- HTTP/2.0 allows a server to “push” content to a client before the client asks for it while for HTTP/1.0 a server only serves content to a client device if the client asks for it, now this may sound like unnecessary overhead work in HTTP/2.0 however the content pushed without requesting is the content which client must request.



Server Push

**Header compression**- 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 making the packets smaller in size resulting in faster loading of data.



HPACK

A new version HTTP/3.0 is also released in 2018, although it’s not that popular or vastly adopted till date.

**2.Objects and its representation in JavaScript.**

Objects are the representation of real-world entities in any language representing things by defining its properties along with their values. In JavaScript, objects may be defined as an unordered collection of related data, of primitive or reference types, in the form of **“key: value”** pairs.

W**ays to create an object in JavaScript are as follows-**

**Object literal**

object literal is a comma-separated list of name-value pairs wrapped in curly braces. Object literals encapsulate data, enclosing it in a tidy package.

*var car={id:1 , name:’abc’ , display:function() }*

As evident from the above example property values can be of any data type, including array literals, functions, nested object literals, or primitive data type.

**Object.create()**

The method creates a new object, using an existing object as the prototype of the newly created object.

*using the object literal example as prototype-*

*var car2 = Object.create(car);*

*car.id=2;*

*car.name=’xyz’;*

**Object constructor**

Useful when we require to create multiple objects of similar type. In this case, a constructor (kind of blueprint) is created and multiple objects can be initialized using the new keyword using the constructor as a wrapper for the newly created objects.

*construction function-*

*function Person(name, age, eye) {  
this.Name = name;  
this.age = age;  
this.eyeColor = eye;  
}*

*creating objects using constructor-*

*var p1= new Person(“John”, 50, “blue”);  
var p2= new Person(“Sally”, 48, “green”);*

**Object.assign()**

It is used to copy the values and properties from one or more source objects to a target object. It invokes getters and setters since it uses both [[Get]] on the source and [[Set]] on the target.

*Here is an example where properties from three source objects are getting assigned to a new object using Object.assign()*

*Input : var obj1 = { a: 10 };  
var obj2 = { b: 20 };  
var obj3 = { c: 30 };  
var new\_obj = Object.assign(o1, o2, o3);  
console.log(new\_obj);  
Output : Object { a: 10, b: 20, c: 30 }*

**Object.fromEntries**

This method transforms a list of key-value pairs into an object.

*const entries = new car([  
[‘id’, 4],  
[‘color’, ‘blue’]  
]);*

*const car1= Object.fromEntries(entries);*

*console.log(car1);  
output: Object { id: 4, color: ‘blue’}*

*Unlike other object-oriented programming languages, javascript doesn’t have classes instead of that javascript is a prototype-based language allowing all the functionalities as in other class-based programming languages like JavaScript allows you to create hierarchies of objects and to have the inheritance of properties and their values and all this is done mainly using the constructor functions.*