# Difference between HTTP1.1 vs HTTP2

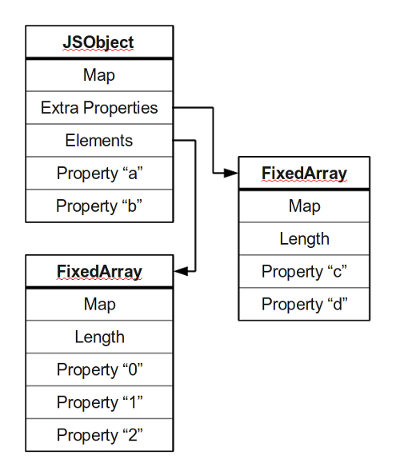
* **Multiplexing:** HTTP/1.1 loads resource’s one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. In contrast, HTTP/2 is able to use a single [TCP](https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/) connection to send multiple streams of data at once so that no one resource blocks any other resource. 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 web pages, 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. The server also sends a message letting the client know what pushed content to expect
* **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.

# **objects and their internal representation in Javascript**

Objects are the representation of real-world entities in any language representing things by defining their 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.

A JavaScript object has properties associated with it. A property of an object can be explained as a variable that is attached to the object. Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects. The properties of an object define the characteristics of the object. We can access the properties of an object with a simple dot-notation:

*objectName.propertyName*

Like all JavaScript variables, both the object name (which could be a normal variable) and property name are case-sensitive. You can define a property by assigning it a value.

Most objects contain all their properties in a single block of memory *(‘a’ and ‘b’)*. All blocks of memory have a pointer to a map, which describes their structure.

Most objects contain all their properties in a single block of memory (‘a’ and ‘b’). All blocks of memory have a pointer to a map, which describes their structure.

Numbered properties are stored separately, usually in a contiguous array.

The JavaScript standard allows developers to define objects in a very flexible way, and it is hard to come up with an efficient representation that works for everything. An object is essentially a collection of properties: basically key-value pairs. We can access properties using two different kinds of expressions:

obj.prop

obj[“prop”]

Numbered properties are stored separately, usually in a contiguous array.

The JavaScript standard allows developers to define objects in a very flexible way, and it is hard to come up with an efficient representation that works for everything. An object is essentially a

collection of *properties*: basically key-value pairs. We can access properties using two different kinds of expressions:

Obj.propobj[“prop”]