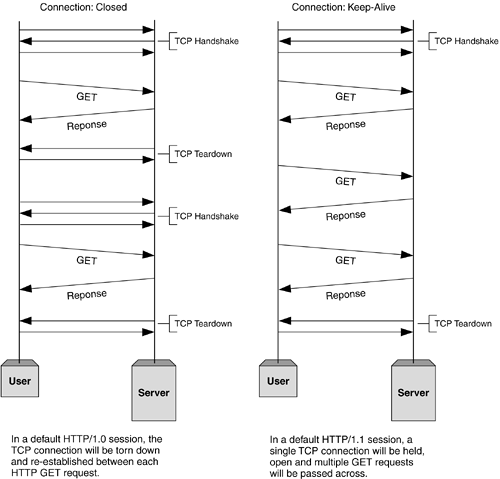
**DIFFERENCE BETWEEN HTTP1.1 vs HTTP2**

**HTTP 1.1**

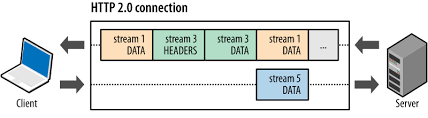
HTTP 1.1 is the latest version of Hypertext Transfer Protocol (HTTP), the World Wide Web application protocol that runs on top of the Internet's TCP/IP suite of protocols. HTTP 1.1 provides faster delivery of Web pages than the original HTTP and reduces Web traffic. HTTP 1.1 is supported by the latest Web servers and browsers. The first version of HTTP, referred to as HTTP/0.9, was a simple protocol for raw data transfer across the Internet. HTTP/1.0 improved the protocol by allowing messages to be in the format of MIME-like messages, containing meta information about the data transferred and modifiers on the request/response semantics. However, HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, or virtual hosts. In addition, the proliferation of incompletely-implemented applications calling themselves “HTTP/1.0” has necessitated a protocol version change in order for two communicating applications to determine each other’s true capabilities.



* Instead of opening and closing a connection for each application request, HTTP 1.1 provides a persistent connection that allows multiple requests to be batched or pipelined to an output buffer.
* The underlying Transmission Control Protocol layer can put multiple requests (and responses to requests) into one TCP segment that gets forwarded to the Internet Protocol layer for packet transmission. Because the number of connection and disconnection requests for a sequence of "get a file" requests is reduced, fewer packets need to flow across the Internet. Since requests are pipelined, TCP segments are more efficient.
* Overall result is less Internet traffic and faster performance for the user. Persistent connection is similar to Netscape's HTTP 1.0 extension called **KeepAlive**, but provides better handling of requests that go through proxy servers.
* When a browser supporting HTTP 1.1 indicates it can decompress HTML files, a server will compress them for transport across the Internet, providing a substantial aggregate savings in the amount of data that has to be transmitted.
* Image files are already in a compressed format so this improvement applies only to HTML and other non-image data types.
* HTTP 1.1 also provides the ability to have multiple domain names share the same Internet address (IP address). This simplifies processing for Web servers that host a number of Web sites in what is sometimes called virtual hosting.

**HTTP /2**

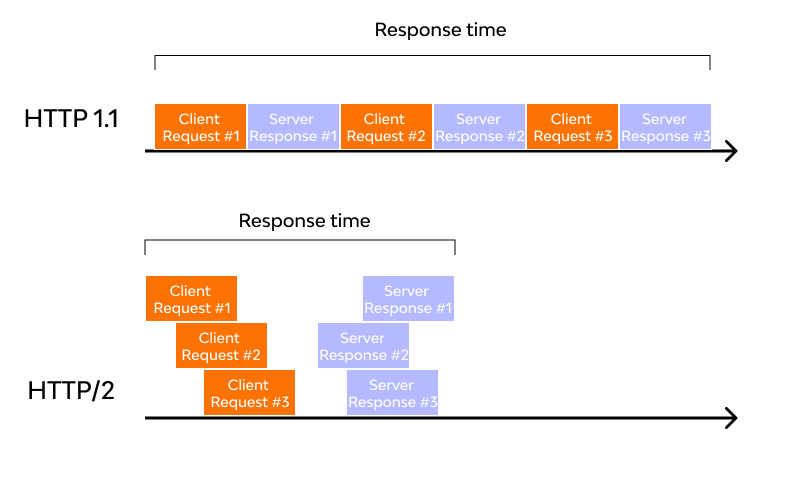
* HTTP/2 will make our applications faster, simpler, and more robust by allowing us to undo many of the HTTP/1.1 workarounds previously done within our applications and address these concerns within the transport layer itself.
* It also opens up a number of entirely new opportunities to optimize our applications and improve performance.
* HTTP/2 does not modify the application semantics of HTTP in any way. All the core concepts, such as HTTP methods, status codes, URIs, and header fields, remain in place. Instead, HTTP/2 modifies how the data is formatted and transported between the client and server, both of which manage the entire process, and hides all the complexity from our applications within the new framing layer and all existing applications can be delivered without modification.



* HTTP/2's primary changes from HTTP/1.1 focus on improved performance. Some key features such as multiplexing, header compression, prioritization and protocol negotiation evolved from work done in an earlier open, but non-standard protocol named SPDY. Chrome has supported SPDY since Chrome 6, but since most of the benefits are present in HTTP/2, it’s time to say goodbye. We plan to remove support for SPDY in early 2016, and to also remove support for the TLS extension named NPN in favor of ALPN in Chrome at the same time. Server developers are strongly encouraged to move to HTTP/2 and ALPN.
* HTTP/2 enables a more efficient use of network resources and a reduced perception of latency by introducing header field compression and allowing multiple concurrent exchanges on the same connection… Specifically, it allows interleaving of request and response messages on the same connection and uses an efficient coding for HTTP header fields. It also allows prioritization of requests, letting more important requests complete more quickly, further improving performance.



|  |  |
| --- | --- |
| HTTP 1.1 | HTTP/2 |
| Works on Textual Format | Works on binary protocol |
| Blocks all requests behind it until it doesn’t get its all resources | Allows multiplexing so one TCP connection is required for multiple requests |
| Uses requests resource inlining for getting multiple pages | Uses PUSH frame by server that collect all multiple pages |
| Compresses data by itself | Uses HPACK for data compression |
| It is relatively secure since it uses digest authentication, NTLM authentication. | 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. |
| Expands on the caching support by using additional headers like cache-control, conditional headers like If-Match and by using entity tags. | 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/1.1 provides faster delivery of web pages and reduces web traffic as compared to HTTP/1.0. However, TCP starts slowly and with domain sharding (resources can be downloaded simultaneously by using multiple domains), connection reuse and pipelining, there is an increased risk of network congestion. | 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 REPRESENTAION IN JAVASCRIPT**

Objects, in JavaScript, are the most important data type and form the building blocks for modern JavaScript. These objects are quite different from JavaScript’s primitive data types (**Number, String, Boolean, null, undefined, and symbol**) in the sense that these primitive data types all store a single value each (depending on their types).

**SYNTAX:**

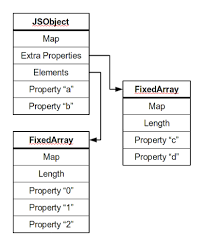
let object\_name = {

key\_name : value,

...

}

* Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.
* An object is a reference data type. Variables that are assigned a reference value are given a reference or a pointer to that value. That reference or pointer points to the location in memory where the object is stored. The variables don’t actually store the value.
* 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.
* An object can be created with figure brackets {…} with an optional list of properties. A property is a “**key: value**” pair, where a key is a string (also called a “property name”), and the value can be anything.



**Inherited Properties**

Inherited properties of an object are those properties that have been inherited from the object’s prototype, as opposed to being defined for the object itself, which is known as the object’s Own property. To verify if a property is an object’s Own property, we can use the **hasOwnProperty** method.

Property Attributes Data properties in JavaScript have four attributes.

* **value:** The property’s value.
* **writable:** When true, the property’s value can be changed
* **enumerable:** When true, the property can be iterated over by “for-in” enumeration. Otherwise, the property is said to be non-enumerable.
* **configurable:** If false, attempts to delete the property, change the property to be an access-or property, or change its attributes (other than [[Value]], or changing [[Writable]] to false) will fail.

**Creating Objects in JavaScript:**

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

**By object literal:**

The syntax of creating object using object literal is given below:

**Object = {property1:value1,property2:value,…property:valueN};**

Property and value is separated by colon(:)

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

The syntax of creating object directly is given below:

**Var objectName = new Object();**

Here, **new** keyword is used to create object.

**Iterating over all keys of an object:**

To iterate over all existing enumerable keys of an object, we may use **for…in** construct. It is worth noting that this allows us to access only those properties of an object which are enumerable. For instance, properties inherited from the **Object.prototype** are not enumerable. But, enumerable properties inherited from somewhere can also be accessed using the **for…in** construct

**Deleting Properties:**

To Delete a property of an object we can make use of the **delete** operator.