**Difference between HTTP 1.1 and HTTP 2**

**Key Features of HTTP1.0:**

* The concept of headers both for requests (from the client machine) as well as responses (from servers) was introduced. The use of headers such as GET, POST, HEAD added extended flexibility, none of which was possible with the earlier version.
* Version information was now included.
* It allowed a single request/response for every TCP connection.
* Status codes were used to indicate successful requests and to indicate transmission errors.
* The content-type header made it possible to send files other than plain HTML, including scripts and media.

**Key Features of HTTP1.1:**

* It was no longer required for each connection to be terminated immediately after every request was served with a response; instead, with the keep-alive header, it was possible to have persistent connections. It allowed multiple requests/responses per TCP connection.
* The Upgrade header was used to indicate a preference from the client that made it possible to switch to a more preferred protocol if found appropriate by the server.
* HTTP/1.1 provided support for chunk transfers that allowed streaming of content dynamically as chunks and for additional headers to be sent after the message body. This enhancement was particularly useful in cases where values of a field remained unknown until the content had been produced. For example, when the content had to be digitally signed, it was not possible to do so before the entire content gets generated.
* Other features that reinforced its stability were introduced such as:
  + pipelining (the second request is sent before the response to the first is adequately served)
  + content negotiation (an exchange between client and server to determine the media type, it also provides the provision to serve different versions of a resource at the same URI)
  + cache control (used to specify caching policies in both requests and responses)

**Key Features of 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.
* Introduces the concept of multiplexing that interleaves the requests and responses without head-of-line blocking and does so over a single TCP connection.
* It is a binary protocol i.e. only binary commands in the form of 0s and 1s are transmitted over the wire. The binary framing layer divides the message into frames that are segregated based on their type – Data or Header. This feature greatly increases efficiency in terms of security, compression and multiplexing.
* HTTP/2 uses HPACK header compression algorithm that is resilient to attacks like CRIME and utilizes static Huffman encoding.

**Difference between HTTP 1.1 and HTTP 2**

|  |  |
| --- | --- |
| **HTTP 1.1** | **HTTP 2** |
| HTTP1 loads a single request for every TCP connection, | while HTTP2 avoids network delay by using multiplexing. |
| \*it is not possible in HTTP/1.1. | In HTTP/2, developers have hands-on, detailed control over prioritization. This allows them to maximize perceived and actual page load speed |
| HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it | HTTP/2 is able to use a single TCP 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. |

**More about HTTP2**

HTTP/2 offers a feature called weighted prioritization. This allows developers to decide which page resources will load first, every time

In HTTP/2, when a client makes a request for a webpage, the server sends several streams of data to the client at once, instead of sending one thing after another. This method of data delivery is known as multiplexing. Developers can assign each of these data streams a different weighted value, and the value tells the client which data stream to render first.