**Difference between HTTP 1.1 vs HTTP 2**

| **HTTP/1.1** |  |
| --- | --- |
| * It works on the textual format. |  |
| * There is head of line blocking that blocks all the requests behind it until it doesn’t get its all resources. |  |
| * It uses requests resource In lining for use getting multiple pages |  |
| * It compresses data by itself. |  |

* 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 connection to send multiple streams of data at once so that no one resource blocks any other resource.
* Certain resources, like large JavaScript files, may block the rest of the page from loading if they have to load first. More of the page can load at once if these render-blocking resources load last

**HTTP/2**

It works on the binary protocol.

|  |
| --- |
| * It uses PUSH frame by server that collects all multiple pages |
| * HTTP/2 can send multiple requests for data in parallel over a single TCP connection. This is the most advanced feature of the HTTP/2 protocol because it allows you to download web files asynchronously from one server. Most modern browsers limit TCP connections to one server. * HTTP/2 compresses a large number of redundant header frames. It uses the HPACK specification as a simple and secure approach to header compression. Both client and server maintain a list of headers used in previous client-server requests. |

* The latest HTTP version has evolved significantly in terms of capabilities and attributes such as transforming from a text protocol to a binary protocol. HTTP/2 will use binary commands (in 1s and 0s) to execute the same tasks. This attribute eases complications with framing and simplifies implementation of commands that were confusingly intermixed due to commands containing text and optional spaces.
* This capability allows the server to send additional cacheable information to the client that isn’t requested but is anticipated in future requests. For example, if the client requests for the resource X and it is understood that the resource Y is referenced with the requested file, the server can choose to push Y along with X instead of waiting for an appropriate client request.