| **HTTP/1.1** | **HTTP/2** |
| --- | --- |
| It works on the textual format. | It works on the binary protocol. |
| It uses a textual format for requests and responses. This means that the data exchanged between the client and server is in plain text. | It employs a binary format, which is more efficient for parsing and reduces overhead. Binary framing allows for faster communication. |
| There is head of line blocking that blocks all the requests behind it until it doesn’t get its all resources. | It allows multiplexing so one TCP connection is required for multiple requests. |
| Requests and responses are processed sequentially over a single connection. If one resource is slow to load, it can block subsequent requests. | It fully supports multiplexing, allowing multiple requests and responses to be sent concurrently over a single connection. This improves performance by avoiding blocking delays. |
| It uses requests resource Inlining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| Each resource (e.g., HTML, CSS, images) requires a separate TCP connection. Establishing multiple connections can lead to latency. | It can use a single connection for parallelism. This reduces the overhead of establishing multiple connections and improves overall page load times. |
| Headers (metadata) are sent with each request and response. These headers can be verbose and add unnecessary overhead. | It uses header compression, reducing the size of headers and improving efficiency. |
| It compresses data by itself. | It uses HPACK for data compression. |
| The client requests resources one by one. If the server knows which resources are needed next, it can proactively push them to the client. | It introduces server push, allowing the server to send additional resources (e.g., CSS, images) to the client without waiting for explicit requests. This optimizes caching and reduces round trips. |