# HTTP/1.1 VS. HTTP/2
# A TECHNICAL COMPARISON

# 1. Binary vs. Textual Format:

| HTTP/1.1 | HTTP/2 |
| --- | --- |
| Keeps all requests and responses in plain text format. | On the other hand, uses a binary framing layer to encapsulate messages in binary format while still maintaining HTTP semantics (such as verbs, methods, and headers). |

# 2. Multiplexing:

| HTTP/1.1 | HTTP/2 |
| --- | --- |
| Requests and responses sequentially, which can lead to blocking and latency issues. | Fully multiplexed, allowing multiple requests and responses to be sent concurrently over a single connection. This significantly reduces latency and improves performance. |

# 3. Single Connection for Parallelism:

| HTTP/1.1 | HTTP/2 |
| --- | --- |
| Each resource (e.G., HTML, CSS, javascript) requires a separate TCP connection, leading to overhead. | Enables parallelism by using a single connection for multiple resources. This minimizes the overhead associated with establishing multiple connections. |

# 4. Header Compression:

| HTTP/1.1 | HTTP/2 |
| --- | --- |
| Sends headers in plain text with each request and response, resulting in redundant data transmission. | Employs header compression, reducing the overhead caused by headers and improving efficiency |

# 5. Server Push:

HTTP/2 introduces server push, allowing proactive addition of responses into the browser cache. For example, the server can push CSS or JavaScript files to the client before they are explicitly requested, enhancing page load times.

# In summary:

HTTP/2 is faster, more reliable, and better suited for modern web applications. Its multiplexing capabilities, binary format, and header compression contribute to improved performance and reduced latency. As a developer, understanding these differences can guide your decisions when optimizing web applications for the best user experience