HTTP (HYPER TEXT TRANSFER PROTOCOL):

* HTTP, or Hypertext Transfer Protocol, is the underlying protocol used for communication on the World Wide Web. It is an application layer protocol that facilitates the transfer of data between clients and servers. HTTP defines a set of rules for how messages are formatted and transmitted, as well as how web servers and browsers should respond to various commands.
* HTTP is the foundation of data communication on the web, enabling the retrieval and display of web content. It forms the basis for many higher-level protocols and technologies, including HTTPS (HTTP Secure), which adds a layer of encryption for secure communication.
* Introduced first at 1997 called HTTP0.9 and latest one at 2022 called HTTP3.
* HTTP are often called as VERBS (meaning actions) because it does the same job as verbs.

**KEY FEATURES OF HTTP:**

* **Stateless Protocol:**
* **Client-Server Model:**
* **Request-Response Cycle:**
* **Methods**
* **URLs (Uniform Reform Locators)**
* **Headers**

**DIFFERENCE BETWEEN HTTP1.1 vs HTTP2:**

HTTP/1.1 and HTTP/2 are different versions of the HTTP (Hypertext Transfer Protocol) protocol, which is fundamental to the functioning of the World Wide Web. While HTTP/1.1 has been the dominant protocol for many years, HTTP/2 was introduced to address some of its limitations and improve overall performance. Here are some key differences between HTTP/1.1 and HTTP/2.

* **Multiplexing:**

**HTTP/1.1**: Uses a single, serialized connection per request, which means that only one request-response transaction can occur at a time. This can lead to head-of-line blocking, where a slow-loading resource can delay the loading of subsequent resources.

**HTTP/2:** Introduces multiplexing, allowing multiple requests and responses to be sent in parallel over a single connection. This significantly reduces latency and can improve page load times, especially for complex web pages with numerous resources.

* **Header Compression:**

**HTTP/1.1:** Headers are sent in plaintext with each request and response. While they can be compressed using techniques like gzip, the overhead of sending headers with each request can be significant.

**HTTP/2**: Implements header compression, reducing the overhead associated with sending headers. This can result in lower latency and reduced bandwidth consumption.

* **Binary Protocol:**

**HTTP/1.1**: Uses a plaintext protocol, which is human-readable but can be inefficient in terms of parsing and processing.

**HTTP/2:** Adopts a binary protocol, which is more efficient for machines to parse. While this makes it less human-readable, it enhances performance.

* **Server Push:**

**HTTP/1.1**: Requires multiple round trips to the server to fetch resources referenced in the HTML document. Resources are fetched on-demand.

**HTTP/2:** Introduces server push, allowing the server to push resources to the client before they are explicitly requested. This can reduce the number of round trips and improve page load times**.**

* **Connection Handling:**

**HTTP/1.1:** Requires the opening and closing of multiple connections for concurrent requests, which can be resource-intensive.

**HTTP/2**: Encourages the use of a single, long-lived connection for multiple requests, reducing the overhead associated with opening and closing connections.

| **HTTP/1.1** | **HTTP/2** |
| --- | --- |
| It works on the textual format. | It works on the binary protocol. |
| 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. |
| It uses requests resource Inlining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| It compresses data by itself. | It uses HPACK for data compression. |

**CONCLUSION:**

In summary, while HTTP/1.1 has served the web well for many years, **HTTP/2 introduces improvements in terms of performance, efficiency, and the handling of multiple concurrent requests**. The adoption of HTTP/2 has been driven by the need for faster and more efficient web communication, particularly in the context of an increasing number of complex web applications and the emphasis on optimizing mobile web experiences.