**Brief History of HTTP**

|  |  |
| --- | --- |
| **Year** | **HTTP Version** |
| 1991 | 0.9 |
| 1996 | 1.0 |
| 1997 | 1.1 |
| 2015 | [2.0](https://en.wikipedia.org/wiki/HTTP/2) |
| Draft (2020) | [3.0](https://en.wikipedia.org/wiki/HTTP/3) |

The Hypertext Transfer Protocol (HTTP) is one of the most ubiquitous and widely adopted application protocols on the Internet: it is the common language between clients and servers, enabling the modern web. From its simple beginnings as a single keyword and document path, it has become the protocol of choice not just for browsers, but for virtually every Internet-connected software and hardware application.

**HTTP 0.9: The One-Line Protocol**

The original HTTP proposal by Tim Berners-Lee was designed with *simplicity in mind* as to help with the adoption of his other nascent idea: the World Wide Web. The strategy appears to have worked: aspiring protocol designers, take note.

In 1991, Berners-Lee outlined the motivation for the new protocol and listed several high-level design goals: file transfer functionality, ability to request an index search of a hypertext archive, format negotiation, and an ability to refer the client to another server. To prove the theory in action, a simple prototype was built, which implemented a small subset of the proposed functionality:

* Client request is a single ASCII character string.
* Client request is terminated by a carriage return (CRLF).
* Server response is an ASCII character stream.
* Server response is a hypertext markup language (HTML).
* Connection is terminated after the document transfer is complete.

The request consists of a single line: GET method and the path of the requested document. The response is a single hypertext document—no headers or any other metadata, just the HTML.

From these humble beginnings in 1991, HTTP took on a life of its own and evolved rapidly over the coming years. Let us quickly recap the features of HTTP 0.9:

* Client-server, request-response protocol.
* ASCII protocol, running over a TCP/IP link.
* Designed to transfer hypertext documents (HTML).
* The connection between server and client is closed after every request.

**HTTP/1.0: Rapid Growth and Informational RFC**

The period from 1991 to 1995 is one of rapid coevolution of the HTML specification, a new breed of software known as a “web browser,” and the emergence and quick growth of the consumer-oriented public Internet infrastructure.

**THE PERFECT STORM: INTERNET BOOM OF THE EARLY 1990S**

From this period of rapid experimentation, a set of best practices and common patterns began to emerge, and in May 1996 the HTTP Working Group (HTTP-WG) published RFC 1945, which documented the “common usage” of the many HTTP/1.0 implementations found in the wild. Having said that, an example HTTP/1.0 request should look very familiar:

* Request line with HTTP version number, followed by request headers
* Response status, followed by response headers

The preceding exchange is not an exhaustive list of HTTP/1.0 capabilities, but it does illustrate some of the key protocol changes:

* Request may consist of multiple newline separated header fields.
* Response object is prefixed with a response status line.
* Response object has its own set of newline separated header fields.
* Response object is not limited to hypertext.
* The connection between server and client is closed after every request.

**HTTP/1.1: Internet Standard**

The work on turning HTTP into an official IETF Internet standard proceeded in parallel with the documentation effort around HTTP/1.0 and happened over a period of roughly four years: between 1995 and 1999. In fact, the first official HTTP/1.1 standard is defined in RFC 2068, which was officially released in January 1997, roughly six months after the publication of HTTP/1.0. Then, two and a half years later, in June of 1999, a number of improvements and updates were incorporated into the standard and were released as RFC 2616.

The HTTP/1.1 standard resolved a lot of the protocol ambiguities found in earlier versions and introduced a number of critical performance optimizations: keepalive connections, chunked encoding transfers, byte-range requests, additional caching mechanisms, transfer encodings, and request pipelining.

With these capabilities in place, we can now inspect a typical HTTP/1.1 session as performed by any modern HTTP browser and client:

* Request for HTML file, with encoding, charset, and cookie metadata
* Chunked response for original HTML request
* Number of octets in the chunk expressed as an ASCII hexadecimal number (256 bytes)
* End of chunked stream response
* Request for an icon file made on same TCP connection
* Inform server that the connection will not be reused
* Icon response, followed by connection close

HTTP/1.1 changed the semantics of the HTTP protocol to use connection keepalive by default. Meaning, unless told otherwise (via Connection: close header), the server should keep the connection open by default.

**HTTP/2: Improving Transport Performance**

The primary focus of HTTP/2 is on improving transport performance and enabling both lower latency and higher throughput. The major version increment sounds like a big step, which it is and will be as far as performance is concerned, but it is important to note that none of the high-level protocol semantics are affected: all HTTP headers, values, and use cases are the same.

Any existing website or application can and will be delivered over HTTP/2 without modification: you do not need to modify your application markup to take advantage of HTTP/2. The HTTP servers will have to speak HTTP/2, but that should be a transparent upgrade for the majority of users. The only difference if the working group meets its goal, should be that our applications are delivered with lower latency and better utilization of the network link!