**HTTP 1.1**

* HTTP/1.1 practically allows only one outstanding request per TCP connection (though [HTTP pipelining](https://en.wikipedia.org/wiki/HTTP_pipelining) allows more than one outstanding request, it still doesn’t solve the problem completely).
* The other problem with HTTP/1.1 is the duplication of data across requests (cookies and other headers).
* Implementation of multiple parallel TCP connections to every domain (the number of parallel connections varied per browser). But this head-of-line blocking nature of HTTP/1.1 is a major bottleneck for faster loading applications.
* The other problem with HTTP/1.1 is the duplication of data across requests (cookies and other headers). Too many requests means too much redundant data, which would impact performance.
* This led to the development of techniques like image sprites (combining multiple image requests into a single one) and domain-sharding (splitting the requests for resources over multiple domains to increase the number of possible parallel TCP connections).

**HTTP/2**

* Allows using same TCP connection for multiple parallel requests.
* HTTP/2 (originally named HTTP/2.0) is a major revision of the [HTTP](https://en.wikipedia.org/wiki/HTTP) network protocol used by the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web). It was derived from the earlier experimental [SPDY](https://en.wikipedia.org/wiki/SPDY) protocol, originally developed by [Google](https://en.wikipedia.org/wiki/Google). HTTP/2 was developed by the HTTP Working Group (also called httpbis, where "[bis](https://en.wiktionary.org/wiki/bis)" means "second") of the Internet Engineering.
* HTTP/2 provides an optimized transport for HTTP semantics. HTTP/2 supports all of the core features of HTTP/1.1 but aims to be more efficient in several ways.
* HTTP/2 adds a new interaction mode whereby a server can push responses to a client.

I would also like to present the example how the http1.1 works and http2 works from the following link <https://imagekit.io/demo/http2-vs-http1>

**GET and POST**

**GET**

* HTTP protocol supports several request methods you can use while sending requests using HTTP or HTTPS protocol. GET is one of them. As the name suggests the GET method is to retrieve a page from the HTTP Server. You can identify a GET request by looking method attribute on the HTTP Request part.
* One important property of GET request is that any request parameter or query parameter is passed as [URL encoded string](http://javarevisited.blogspot.com/2012/01/url-rewriting-url-encoding-in-servlet.html), appended using "?" character which makes it non-secure because whatever information you pass in URL String is visible to everybody.
* There is no side effect of repeated request. for example clicking a link which points to another page. it doesn't matter if you click the link twice or thrice , This also gives chance browser of the server to catch the response for faster retrieval.

**POST METHOD**

* POST in HTTP request. In POST method data is not sent as part of a URL string to server instead in POST, data is sent as part of message body.
* POST method is also used for submitting information to server, any information which can alter state of application like adding item into shopping cart, making payments etc.

**DIFFERENCE**

* GET method passes request parameter in URL String while POST method passes the request parameter in request body.
* GET request can only pass limited amount of data while POST method can pass large amount of data to server.
* GET requests can be bookmarked and cached unlike POST requests.
* GET is mostly used for view purpose (e.g. SQL SELECT) while POST is mainly use for update purpose (e.g. SQL INSERT or UPDATE).