1. **Basics of HTTP**

* HTTP stands for **Hyper Text Transfer** **Protocol**. Standard application-level protocol used for exchanging files on the World Wide Web(WWW). HTTP runs on top of the TCP/IP protocol.
* Web browsers are HTTP clients that send file requests to Web servers, which in turn handle the requests via an HTTP service.
* The HTTP is meant **for request/response** depending on a client-server architecture where the user requests information through a web browser to the web server, which then responds to the requested data.
* HTTP is **stateless**, which means there is no connection among two requests being consecutively carried out on the same connection. However, when the core of HTTP is itself a stateless one, HTTP cookies provide in making use of **stateful sessions.** Through the concept of header extensibility, HTTP cookies can be incorporated into the workflow, making session creation on each HTTP request for sharing the same content.

1. **HTTP Request and Response**

Steps involved in client and browser communication:

* A client browser sends an HTTP request to the web server
* A web server receives the request
* The server runs an application to process the request
* The server returns an HTTP response to the browser
* The client receives the response

1. **HTTP1.1 Features**

* **Proxy support and the Host field:**
* HTTP 1.1 has a required Host header by spec.
* This header is useful because it allows you to route a message through proxy servers, and also because your web server can distinguish between different sites on the same server.
* **Persistent connections:**
  + - HTTP 1.1 also allows you to have persistent connections which means that you can have more than one request/response on the same HTTP connection.
* **Caching**:
* HTTP 1.1 expands on the caching support a lot by using something called 'entity tag'. If 2 resources are the same, then they will have the same entity tags.
* Supports persistent and pipelined connections
* Supports chunked transfers, compression/decompression
* Supports virtual hosting (a server with a single IP Address hosting multiple domains)
* Supports multiple languages
* Supports byte-range transfers; useful for resuming interrupted data transfers

1. **HTTP2 Features**

* **Binary**: Commands use 1s and 0s and not text
* **Multiplex**: Permits multiple requests and responses to be sent at the same time
* **Compression**: Compresses headers that have been requested previously to make things more efficient
* **Stream prioritization**: This allows for the exchange of successive streams at one time
* **Server push:** The server can send additional information needed for a request before it is requested
* **Increased security**: HTTP/2 is supported through encrypted connections

1. **HTTP1.1 vs HTTP2**

* **Compression**:
* HTTP/1.1 does not compress headers by default. Wehere as HTTP/2 offers built-in compression of the request headers
* Web applications usually accept a range of different headers, such as authorization, caching directives, and client information. While compression of these might not make much of a difference for a single request, there is a lot of data sent over the network to be saved when compressing them in high-traffic applications.
* **Binary protocol:**
* HTTP/2 is binary instead of textual as HTTP/1.1. This means simplified implementation of commands that previously could be mixed up due to optional whitespace when using the text format. Browsers that support HTTP/2 will convert textual commands into binary before sending them over the network.
* **Security:**
* Because of the binary format used by HTTP/2, there is no longer a risk with response splitting attacks that are possible with HTTP/1.1.
* **Performance & Delivery models:**
* While the HTTP/1.1 protocol delivers responses based on a single request, HTTP/2 uses multiplexing and server push features to increase the delivery performance.
* **Buffer overflow:**
* In HTTP/1.1, the flow control used to manage the available buffer space is implemented at the transport layer. In HTTP/2, the client and server can implement their own flow controls to communicate the available buffer space.
* **Multiplexing:**
* HTTP/2 enables full request and response multiplexing. This means a connection made to a web server from your browser can be used to send multiple requests and receive multiple responses. This gets rid of a lot of the additional time that it takes to establish a new connection for each request. HTTP/1.1 does not support multiplexing.
* **Faster encrypted connections:**
* HTTP/2 uses the new ALPN extension, which allows for faster-encrypted connections since the application protocol is determined during the initial connection. Using HTTP/1.1 without ALPN needs additional round trips for the encryption handshake.