Communicating over HTTP requires at least two parts: the client who requests some information and the server who responds to this request and serves the content. HTTPS is the secured version of HTTP, which makes it better suited to instances where security is a concern. But let's find out how this works. With HTTPS, the client's computer encrypts data before it starts its journey to the server. The server then decrypts this client-side data and processes it. But the web server also encrypts the response data and sends the encrypted content back to your browser. Your browser then decrypts the response and displays it. Since the content is encrypted, it is more secure and very difficult to steal or retrieve information from it.

**HTTP methods**

There are five methods commonly used when accessing content over HTTP. They are: GET, which retrieves a resource; POST, used to send data to the server and then create a record; PUT, which updates the whole resource; PATCH, which are used to partially update a resource; and DELETE, which deletes a resource.

**HTTP requests**

HTTP version type , URL , method , request headers , body(optional)

what are the HTTP request headers and body again? Well, when you submit a form like your username and password to sign into a website, that data is passed to the web server as the HTTP body as either a raw JSON string or a form URL encoded string. HTTP headers, on the other hand, are a core part of every HTTP request and may contain extra information that helps the server make some decisions on how to present the content. Examples of HTTP request headers are cookies, user agents, and referrers. After an HTTP request, comes the HTTP response, which consists of information that the browser uses to properly display the content and the response body.

**HTTP response**

Requested resource

Content length

Content type

Headers

Also the response content

ETags

Time last modified

HTTP status codes

**KISS | Keep, It, Simple, Stupid**

Filter, order, paginate

Version

Cache

Rate Limit

Monitor