1: HTTP

Defintion

ANALOGY

Communication on the web is similar to communication through post/mail. If I have to send my friend a long message, I will have to use a letter. If I have to send a short message, then I'll use a post card. Hence, depending upon my requirement, I have to follow a certain set of rules/protocols to complete the task.

Web communication is similar. One needs to follow certain protocols depending upon the messages/data that needs to be transferred. The most common being: **HTTP (Hyper text transfer protocol)**, FTP (file transfer protocol) and SMTP (Simple mail transfer protocol) etc.

Our focus is HTTP.

HTTP is a protocol followed to communicate between 2 machines/programs (Client and Server). The communication can be in the form of a text, binary, photos, HTML pages etc.

Why do we need HTTP?

2 words: Seamless communication. (Like grammar defines the rules of a language).

How HTTP works?

Ans. Request and Response cycle between a web client and a web server.

A Web servers is like a repository of resources called web resources. Almost everything we see while surfing eg. News articles, videos, images can be found on a server. A client needs to access these.

- Step 1: I open a web browser
- Step 2: I search for a URL: http://www.MyTraining.com/Writing.html
- Step 3: My web browser sends a request to server which hosts 'My training'
- Step 3.1: Within 'MyTraining' get me the document called 'Writing.html'.

- Step 4: The server finds and sends the document.

Breaking down a URL

1. Full form: Uniform Resource Locator
2. Why is it used? To get the location of a resource on the internet.
3. A URL has **3 primary** components:
    - Scheme/Protocol to be used to access the resource: eg. HTTP protocol
    - Address of server on the internet (aka Server's IP address) eg. www.MyTraining.com
    - The path to be followed within the server to find the resource eg. /Writing.html

To note: There are other components to a URL. The other most important is Port number. Port number tells about the application/service within the IP. Eg. Going to a post office using IP address and port number is the counter number.

URL components (Expand)

HTTP messages

1. Syntax: Start line, Header and Body.
2. The status codes: 1xx, 2xx, 3xx, 4xx, 5xx
    a. 1xx: Informational* (need more understanding)
    b. 2xx: OK
    c. 3xx: Redirect
    d. 4xx: Client side error
    e. 5xx: Server side error
3. **Format of request message:**

```
<method> <request-URL> <version>
    <headers>

    <entity-body>
```

4. **Format of a response message:**

```
<version> <status> <reason-phrase>
     <headers>

     <entity-body>
```

5. **Summarising HTTP messages**

HTTP components	Explanation
Version	Version numbers provide applications speaking HTTP with a clue about each other's capabilities and the format of the message.
Method	In a request line, method tells server what to do. examples of method are **GET :** Get documents/files etc. **PUT:** For new info or updation of info eg. update inventory **POST:** Add new information on server. eg. creating new account **DELETE:** Delete data on server
Status codes	Server tells client what happened
Reason phrases	It provides a textual explanation to status code.
Headers	Header fields add additional information to request and response messages. They are NAME:VALUE PAIRS. eg. Content length: 19 Classification 1. **General headers:** Appear in both request and response messages eg. Date, time. ◦ `1Date: Tue, 3 Oct 1974 02:16:00 GMT` 2. **Request headers**: Gives server more information about client's capabilities, format requirements etc. ◦ `1Accept: */*` 3. **Response headers:** Gives client more information about servers they are interacting with ◦ `1Server: Tiki-Hut/1.0`

Other concepts to remember:

1. **TCP/IP:** HTTP messages are sent through TCP/IP. Think of them like a tunnel through which the message goes from one end to another. TCP parses the data into small packets, puts a no. on them called IP number and sends through the tunnel.

Click here to expand...

1. **Proxies:** Special servers that sit between clients and servers. They get request from clients and forward it to actual servers (perhaps after modifying the request). Mostly used for security.

2. **Gateways:** Used to convert HTTP traffic to some other protocol. For eg. HTTP to FTP. act as intermediary servers.
3. **Cache:**

Analogy: How is Zepto able to deliver groceries within 10 minutes? By opening dark stores closer to customers. Similarly, A web cache is a special HTML proxy server that keeps most frequented documents closer to clients.