1st

Codedamn Frontend Accelerator.

⟹ Goals :-

- How domain works
- Role of DNS
- port numbers and Http ports
- writing raw socket request
- fundamental understanding of Http.

→ internet and DNS

* What is DNS ?
  DNS stands for domain Name System, URL's that we use to access sites doesnt make sense to computers.

- Simple task of domain name is to convert the url string into numbers that the computer can understand.
- this number is called as Ip address / internet protocol address.

- DNS is a system/technology which allows computers to resolve host names.

DNS (google.com ⟶ 65.312.4.16)

- DNS is actually the phone book of the internet, each device connected to the internet has a unique IP address, which other machines use to find the device.

* How does DNS work ?

When a user wants to load a webpage, a translation must occur between what a user types into their web browser (example.com) & the machine friendly address necessary to locate the example.com webpage.

Step 1: Query

When we type example.com into our browser our browser sends a query over the internet to find the website.

a query is a question seeking to look up the question (domain name) and return its corresponding IP address.

the first server our query interacts with is the recursive resolver, this basically knows which other DNS servers it needs to ask to answer your original query.

Step 2:- The root servers.

the first type of DNS server the recursive resolver talks to is called the root server. This knows the DNS information about top level domain such as .com or .net

Step 3: the TLD Name Server.

Each top level domain name server stores DNS information for second level domains (example.com) within the TLD (.com)

When our query reaches the TLD name server, the TLD server answers with the IP address of the domain's name server.

Step 4: The Domain's name server (Hosting Servers)

Next the recursive resolver sends the query to the domain's name server, the domain's DNS server knows the IP address for the full domain (example.com) & the answer is returned to the recursive resolver.

Step 5: The website appears.

Now the recursive resolver knows the IP address for the domain name in our query, the recursive resolver tells the browser what the IP address is,

finally, the browser sends a request to the website to retrive its content, using the IP address it just learnt.

→ Http protocol -

Hyper text transfer protocol, there are certain set of rules which the client & server both have to follow in order to establish a successful communication.

- Http is a plain text protocol, basically the request is in human readable format.

- Status codes are issued by a server in response to a client's request made to the server

✱ Responses are grouped into 5 classes.

- Information response (100 - 199)
  An information response indicates the request was received and understood.
  It is issued on provisional basis while request process continues, it alerts the client to wait for final response.

- Successful Response (200 - 299)
  These classe of status codes indicates the action requested by the client was received, understood and accepted.

- Redirection messages (300 - 399)
  These status codes indicate the client must take additional action to complete the request. Many of these are used in URL redirection.

- Client Error Response (400 - 499)
  These status codes are intended for situation in which the error seems to have been caused by the client.

- Server Error Response (500 - 599)
  The Server failed to fulfil a Request

* Port Number.
File transfer protocol (ftp) is a standard communication protocol used for file transfers between server & a client on a computer network.

- port number is used to differentiate the service request that's coming in to the server.

- by default http is assigned port no. 80. https is assigned port no. 443.

* Http Request Methods :-
Http defines a set of request methods to indicate the desired action to be performed for a given resource.

- Get :-
  request's a representation of the specified resource, Request's using get should only retrieve data.
  Eg browsing.

- Head:-
  the head method asks for a response identical to get request, but without the response body.

- Post :-
  the post method submits an entity to the specified resource, often causing a chang in state or side effects on the server
  Eg login form. (send input data for processing)

- Put :-
  the put method is used to transfer data to the server with the request payload. ie deposits data onto the server, inverse of get.

- Patch :-
  partially modify the resource.

Note post acts like a universal request and can be used in any way a user intends. because it supports a body ~~sor~~ as a pay load.

* Http Request Headers.
  Request Header is an Http header that can be used in an Http request to provide information about the request context, so that the server can tailor the response.

Note Net Cat is basically a utility which allows you to open tcp and udp connections to different services.

⇒ Html5 / Css 3.

→ Html Basics.

* comments in html.
  // → for single line
  <-- --> → for a comment block

* Html tags:-
  html tags are like Keywords which defines that how web browser will format and display the content.
  Html tags contain 3 main parts; opening tag, content and closing tag. (some tags are unclosed)

• When a browser reads a HTML document, it reads it from top to bottom & left to right, these tags are then used to create html documents and render their properties, each tag has its own properties.

- most used tags:

x	<html>	</html>	— root element
x	<head>	</head>	— document head
x	<title>	</title>	— page title
x	<body>	</body>	— page's content
x	<H1>	</H1>	— A section heading
x	<p>	</p>	— A paragraph
x	<a>	</a>	— A link
x	<img>	</img>	— An image
x	<div>	</div>	— block level container for content
x	<span>	</span>	— An inline container for content.

- inline vs block

block tags are the tags that consume the entire ~~length~~ width of the page.
inline tags consumes JUST the width of the content that it holds.

Eg:- block → <p>, <div>, <H1>
inline → <span>, <a>, <em>, <strong>