What happens when you type a URL into your Browser ??

What constitutes a URL ?

Human-readable way to know what we are interested in

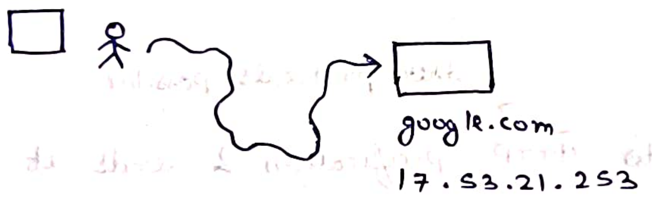
Optional key-value pairs we can pass to additional optional info in the request

$$\underbrace{\text{https}}_{\text{scheme}}\text{://}\overbrace{\text{www.google.com}}^{\text{domain}}\text{/}\underbrace{\text{api/search}}_{\text{path}}\text{ ? }\overbrace{q = \text{home}}$$

Tells browser which protocol to use while connecting.
other schemes : http, ws, etc.

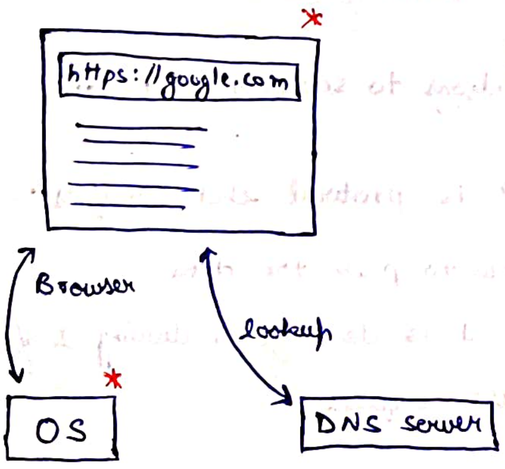
on the product we are accessing, I am interested in this path / resource

The DNS Resolution



Every machine on the internet has an "address" enabling us to reach it over the network
↳ this is IP address

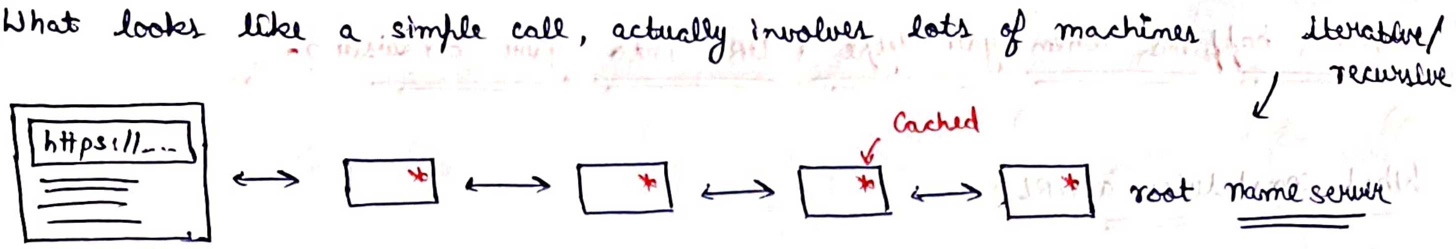
Easier to remember "google.com" than the IP address. Hence we need a way that converts google.com ⟶ 17.53.21.253



Browser does a DNS lookup to get the associated IP address

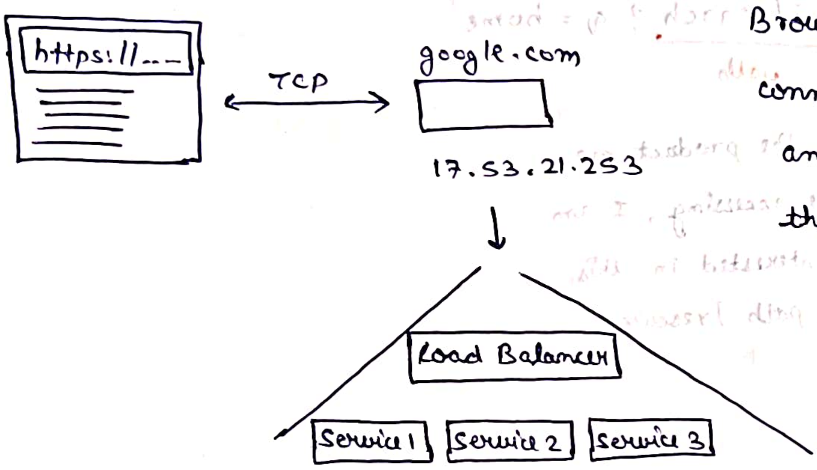
DNS information is heavily cached in the browser, in the OS, across all machines of DNS resolution

What looks like a simple call, actually involves lots of machines



After this resolution process, browser has the IP address to connect to.

Establishing the Connection



Browser now establishes a TCP connection with the machine (server) and can now talk to it over the network.

This in itself is a Pandora's Box & constitutes of 1000s of machines

Sending the Request

other protocols possible

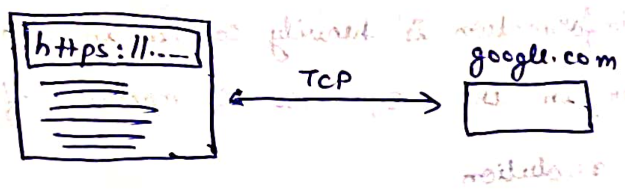
Browser now compiles the request into HTTP specification & sends it across to the server.

given we are just hitting URL in browser it fires an HTTP GET to the server.

```
GET   /api/search?q=home   HTTP/1.1
Host: www.google.com
Connection: keep-alive
```

Headers

instructions to server + metadata



HTTP is protocol that specifies :-

① how to pack the data

② what to do before, during & after the request.

Server processes the request

Once the server receives the HTTP request, it parses the above message & understands what needs to be done.

Server may just load the file from local disk & serve
it may make call to DB to get response
it may throw error if malformed

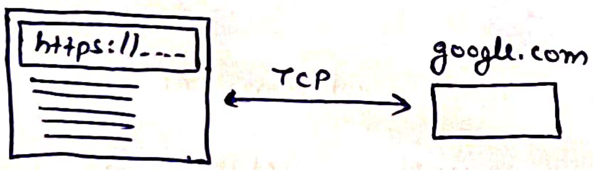
It compiles a proper HTTP response & respond back over same TCP conn.

```
HTTP/1.1  200  OK
Content-Type : text/html
Content-length : 2092
<html> <head> ------
```

Browser upon receiving the response

HTTP/1.1 200 OK	Status Code
Content-Type : text/html	type
Content-length : 2092	length
<html> <head> --------	body

↓



Browser upon receiving the ~~response~~ http message parses the message, extracts the info and "renders"
↳ showing html as html
text as text, etc.

If Browser does not support the response type then it downloads the file locally.

When HTML is rendered, browser may come across

① linked css files
② img tags to render an image
③ inline Javascript code
↳ starts executing it
(may involve making more HTTP requests)
↳ API calls

↳ it fetches the additional files by going through the exact same process