__/__/____

~~CDC~~

UNDERSTANDING HTTP

1. Stateless

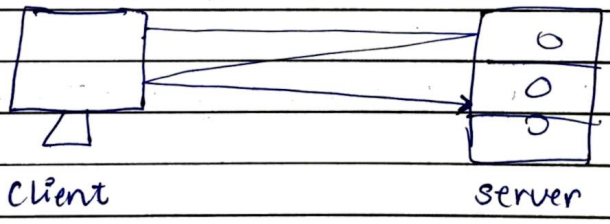
(no memory of past interaction)

↳ each HTTP requests have all the necessary info for the server to process

→ every request is treated as brand new

↳ Stateless system allow each request to be handled independently, which makes them easier to scale horizontally, more fault-tolerant, and better suited for modern cloud and microservices architecture.

2. Client - server model



→ HTTP protocol states that communication is always init by client to get some kind of response by server.

→ HTTP uses uses TCP for connection.

↓

more reliable (connection based)

- HTTP/0.9 (1991) Purpose: Just fetch documents
  - Only support GET
  - No headers
  - No status code
  - Only plain text

Why? The web was just text document.

__/__/____

- ~~The 7 layers of OSI~~ *

→ HTTP/1.0 → each request open a new connection
(1996) ↳ inefficient

→ HTTP/1.1 → purpose : improve performance
(1997)
- Persistant connections (keep alive)
- multiple requests over one connection
- Added caching
- Added Host header

- HTTP/2.0 → purpose : speed & efficiency
(2015)
- Binary protocol (faster then text)
- Multiplexing (multiple requests at once)
- Header compression
- One connection handles everything.

Solved : Head-of-Line blocking in HTTP/1.1

- HTTP/3 purpose : Faster & more reliable
(2022)
- Runs on QUIC instead of TCP
- Uses UDP
- faster connection setup
- better performance on poor networks (mobile)

Why it exists : TCP was bottleneck.

1. HTTP is stateless
2. HTTP/1.1 is still everywhere
3. HTTP/2 improve performance
4. HTTP/3 improve reliability on bad networks.
5. HTTPS = HTTP + encryption

__/__/____

Types of HTTP headers

- Request Headers
  - User - Agent
  - Authorisation
  - Cookie
  - Accept

→ Request headers helps server understand the clients environment, preferences and its capabalities.

- General Header
  - Date
  - Cache - control
  - Connection

- Representation Headers
  - Content - Type
  - content - Length
  - content - Encoding
  - Etag

- Security Header
  - Strict - Transport - Security (HSTS)
  - Content - Security - Policy (CSP)
  - X - Frame - Options
  - X - Content - Type - Options
  - Set - Cookie

__/__/__

- **Extensibility**

→ HTTP is highly extensible because headers can be easily ~~add~~ added or customise without altering the underline protocol.

- **Remote control**

→ HTTP headers acts as kind of RC on the server side. they allow the client to send instructions or preferences to the server influencing how the server responds or processs requests.
e.g, content-type negotiations.

HTTP methods

GET POST PUT (replace the data.) PATCH (use patch unless you have specific use of PUT) DELETE

→ HTTP methods exists to represent different kinds of actions that a client (browser/en) can request on a server. Instead of every request doing the same thing method define the intent of the interaction.

Idempotent vs Non-idempotent

1. **Idempotent**

An operation is idempotent if making the same request multiple times produces the same result on the server.

One request = many requests → same effect

e.g, GET , PUT , HEAD , DELETE , OPTIONS

__/__/____

2. Non-Idempotent

An operation is non-idempotent if repeating the same request changes the result.

One request ≠ many requests

e.g, POST PATCH

OPTIONS method # used in CORS → same origin policy.

→ used to ask the server what operation are allowed on a resource.

What OPTIONS does:

- Returns supported HTTP methods for a URL
- Helps client check capabilities before making real requests.
- Used heavily by browsers for CORS preflight

CORS # (cross-Origin Resource sharing)

↳ is a browser security mechanism that controls which website are allowed to acess resources from another origin.

[CORS decids wheather your browser is allowed to talk to another backend]

• Why CORS exits:

(Browser follow the same-Origin Policy)

- frontend and backend must have the same orgin (scheme + domain + port)
- Otherwise, the browser blocks the response

→ Only browser enforces this

→ Postman / curl ignore CORS

→ Preflight exist for security.

__/__/____

- What is Origin?

  An origin = protocol + domain + port

  e.g,

  https://example.com ✓

  https://example.com:3000 ✗ (different port)

  http://example.com ✗ (different protocol)

- in a cross origin request there two type flows

  ① Simple request ② Preflighted request.

① Simple request

Cross origin request that the browser allows without sending an OPTIONS request first

Conditions:

- Allowed http methods
  - GET
  - POST
  - HEAD

- Allowed headers (only these)
  - Accept
  - Accept language
  - Content-Language
  - Content-type (with restrictions)

- Allowed (content-type)
  - text/plain
  - application/x-www-form-urlencoded
  - multipart/form-data

__/__/____

② Preflighted Request :

is when a browser first sends an OPTION request to ask permission before sending actual request.

- When preflighted happens :

If any these are true

- Method is not GET, POST, HEAD
  e.g, (PUT, DELETE, PATCH)
- Custom headers are used
  (e.g Authorisation)
- Content-type : application/json

- Option request :

An Option request is used to ask a server what is allowed before doing anything risky.

→ The request has a content-type other than application/x-www-form-urlencoded, multipart/form-data or text/pain.

Response header

{ :

Access-control-Max-Age : 86400 ⇒ don't make any preflighted request to me.

→ This will be same for atleast next 24 hours.