WEB WARRIORS

PHASE - ☺

Date |
Page |

Video → **HELLO WORLD** — 25/06/2025

How computer get know about website that searched on a search engine?

Ans-> HTTP - Hyper text transfer protocol

A link that have another link that open another web page

→ Stateless protocol

Every time you visit a page, you are a new user.
No memory of prev. Req.

There is no protocol en Rule in HTTP where State can be maintained.

[THERE are a plethora of Protocals]

Date |
Page |

→ **Sessions**

A session like a class with a teacher, so Teacher only teach to certain Student for that session.

↳ It can store a state of conversation of Browsers and servers.

→ **Cookies**

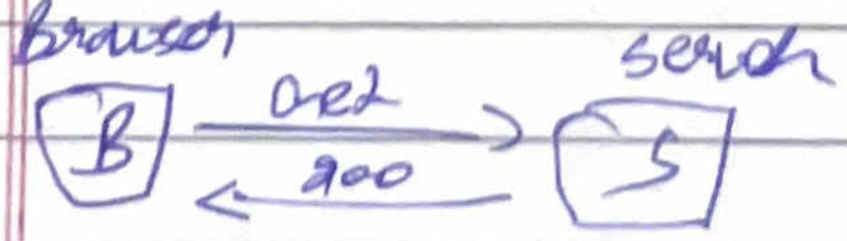
Cookie is a key/value pair. It is just information.

→ **HTTP Header**

Headers also an information. Such as

↳ Client
  ↳ Browser info
  ↳ Date time
  ↳ Cookie to store.

→ **Request Response Model**



Request ⟷ Response

Date /
Page /

Types of Request

GET — to get data from server
POST —
DELETE —

when sending a request from browser REMEMBER

→ what action to perform (Get, Post) → this is called Verb.
→ where to perform (which website to go)
→ was it done? — send a code whether response is ok or not ok.

HTTP/2

~~Not used~~ : HTTP-2 used more now a days. Today.

1) HTTP- means HTTP2
   ↳ Fallback
   HTTP-1.1 is ~~fall back~~ & still used.

2) It uses compression. (Zip - Release size)

(There is not released by us ministry of HTTP 1.0)

3) Encryption (S in HTTP)

EVERY-thing in Computer has Protocols and can be breach.

Kevin
~~Kelvin~~ means C (Wb3)

Every little computation has cost so we don't use Encryption in AWS. Ein internal servers

→ What is TLS and TLS certificate?

TLS - Transport Layer Security
Cert. certificate.

HTTP + TLS → HTTPS
also called TLS certificate and HTTP certificate.

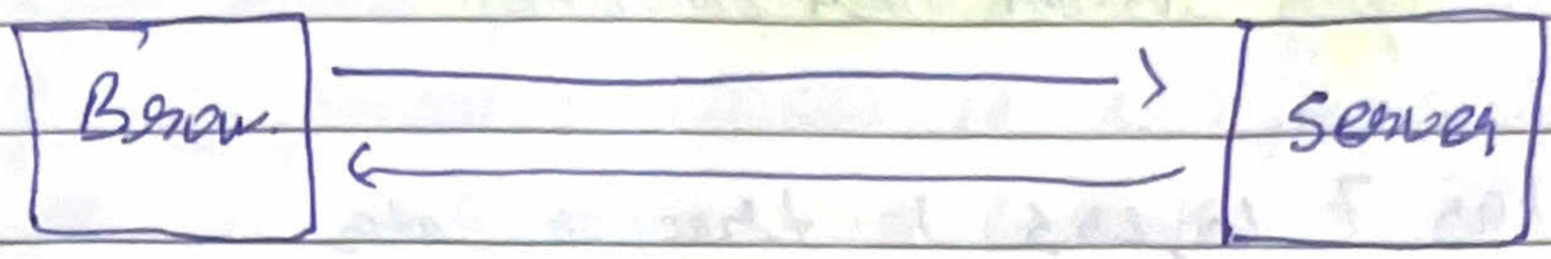
Jargons

1) User Agent - Browser or any code that sent info to server
2) TCP - Transmission Control Protocol (sending)
3) FTP - File Transfer Protocol
4) IP - Internet protocol (name of pc on internet)
5) URL - Uniform Resource Locator
6) DNS - Domain Name Syster (server)
   ↳ points URL to IP
7) Header - (Kiss additional information) - meta data
8) Payload - (Actual data - email, password)
9) Cache - (Store the data) - temp storage

Enhance above more Jargons with ai.

Date |
Page |

Recap



1) Set up TCP connection
2) Exchange TLS certificate
   └ (HTTPS)
3) Send Request verb + URL + Data + (more)
4) Gets the response back with status codes and Data
   (img, csv, text)
5) TCP connection is closed. (State less)

That's the web

---

1) Cookie vs Cache? — cache data temporary.
2) How many ports in computer? — 65,536 ports.

Don't share cookies with anyone

Video-9) Introduction to Web — 27/06/2025

OSI — Open System Interconnection

↳ It has 7 Layers, to transfer data.

1) Physical Layer (works on) - 0-1
2) Data Link Layer (works on the Frame)
3) Network Layer - (~~Router~~ Routing) - IP Addressing
4) Transport Layer - (TCP, UDP)
5) Session Layer
6) Presentation Layer
7) Application Layer

] - Application Layers → servers.

→ What is TCP and UDP

(what is Trade off)

TCP — Transmission Control Protocol.

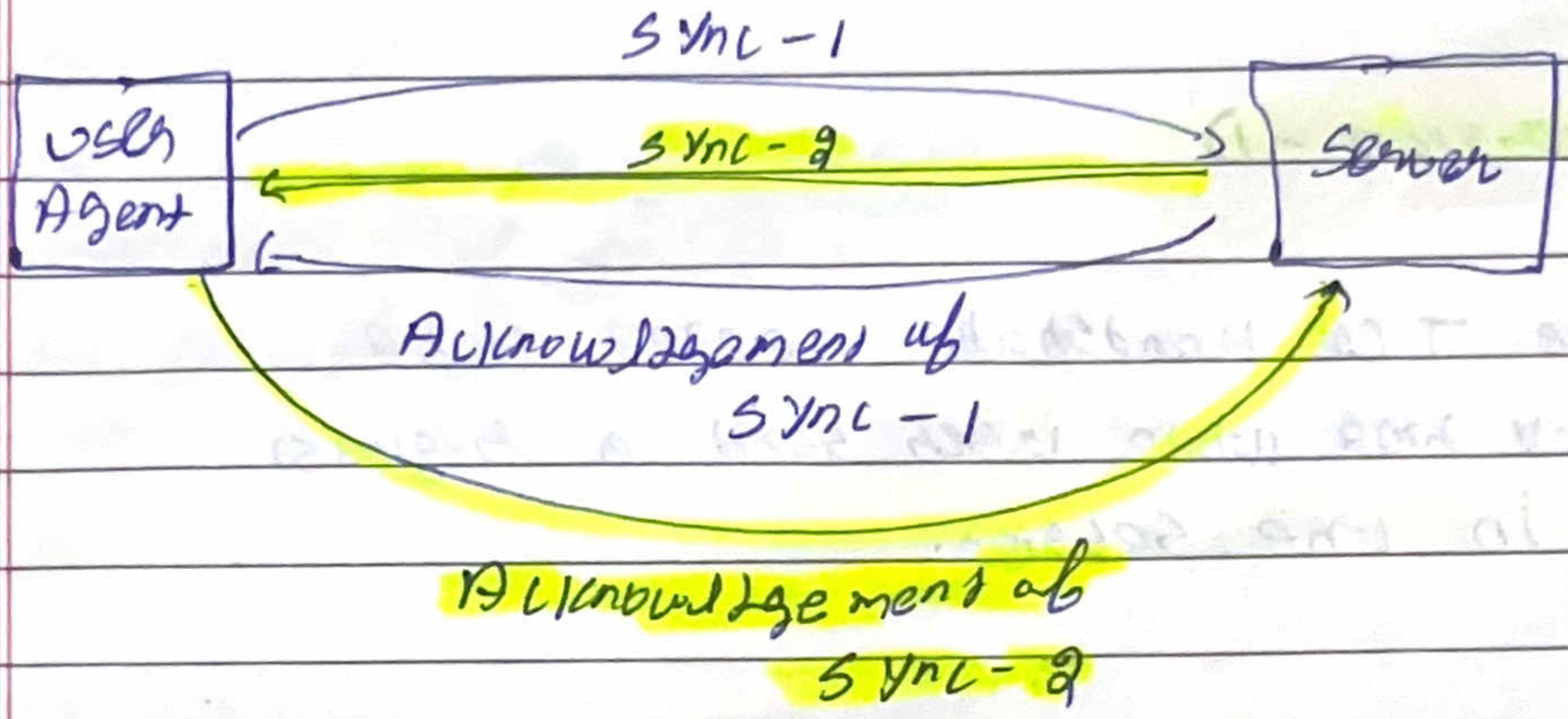
1) Reliable
2) works in ordered way
3) 3-way handshake

Client always have data in chunks also called packets that users transfer to server.

Date |
Page |

TCP

1) ensures every packet is sent.
2) if any packet is missed, it send it again.
3) ~~post~~. it also order the packets.



=) Server send sync-2 and Ack of sync one in a response. (it merge both so send once in spite of two requests)

= Sync — client
↓
SynCK + Ack — server
↓
ACK — client.

After this — o GET, POST begins.

(meaning of offload)

HTTP

Versions

V1 — V1.1 — V2 — V3

V.1 (Version - 1)

↳ Here TCP Handshake needed to be every time when user send a request even in one session.

V.2 (Version - 2)

↳ Here, TCP Handshake require only once to a session.

V.2 - (Version - 2)

↳ It has a feature of Multiplexing which means uses can send multi-types of data.

Overhead — Above your head, Fixed costs in a company.
Latency — How much time it takes to send request.

Date
Page

~~In~~ TCP it takes more time but reliable.

UDP

1) Faster in time
2) not reliable
3) no checkup of packets
4) no order of packets
5) no handshake

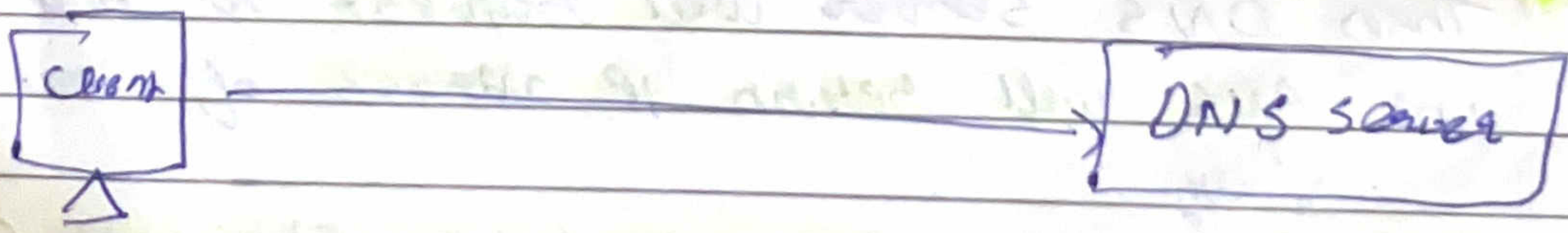
⇒ What happen when type Google.com ? Client.

→ trying to make GET request to use Google.

1) To convert Domain name into an IP address or to get IP address ~~send~~ Browser send request to DNS server.
2) DNS
3) It goes to it's database and return an IP address.

' IT IS CALLED DNS RESOLUTION

How DNS works internally ?



1) DNS send request to ROOT SERVER
   - there are only 13 Root servers.

2) Root name server (Root servers)

   → It return IP address of Top Level Domain servers.

   → TLDs (TOP LEVEL DOMAIN SERVER)
   These are written below!-
   .com, .uk, .ai, .in
   (ڈاٹ کام) (ڈاٹ یو کے) (ڈاٹ ای آئی) (ڈاٹ این)
   These are TLD servers.

3) DNS server will send request to TLD's particular server and then it will return IP address of that specific domain's (Character) ANS (Authoritative Name servers)

   ANS example
   ↳ GoDaddy

4) Then DNS server will request to that ANS and ANS (GoDaddy) will return IP address of that certain website. (Linux Cafe Chai code.com)

   DNS server works on UDP on Port No #53
   USER DATAGRAM PROTOCOL

Date |
Page |

Browser store DNS (IP address) in Cache.

Q) -> what was pipelining ?

Date
Page

Video-3) Git and Github Master class - 23/06/2025

→ VCS → Version Control System

↳ It can maitain history of local code in local machine.

→ Name of some VCS

1) Git
2) Subversion
3) Mercurial
4) Perforce

Git

After initiating Git creates an Git folder called .git that stores every little changes and history.

1) U - Untracked - by default git does not track any file so to track file git add command can be used.

→ git add filename

2) A - Index Added - That certain file will be added for tracking.

Date
Page

3) git add . — it will start tracking all files.

4) There is a compressed file in .git (hidden folder) that is titled index, it is same files that are added to track.

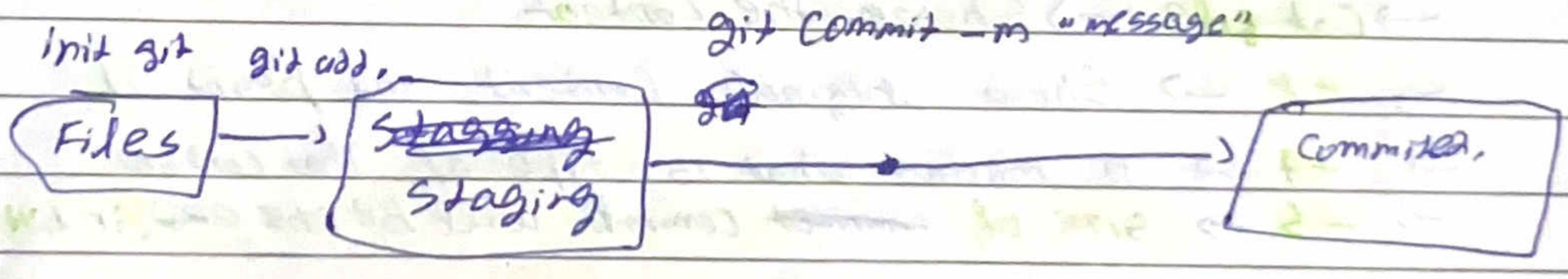
5) to read file command is - cat <filename>

6) Command — git status

↳ It reads git folder and show in a good UI.
If you do not want to track it will show a command to untrack files. It will be removed from index folder.

ALL THESE COMMANDS CAN READ OR WRITE .git FOLDER.

7)



Commit - It is like a check point or snapshot.
- Every commit has ID and hash

8) git log - It check history and shows hash of the commit and every related info.

9) M = modified

10) git diff — It shows what change has been made. Shows difference between new and older version of a specific line.

11) objects folder in .git folder

↳ This folder will have a compressed file that is commited in the folder so every commit will have a diff folder titled first two characters of its hash code.

12) git cat-file -p <hash code of specific commit>

This command will show content of that file at commit.

→ Cat-file → Shows the content
→ -p → Show original content and print it
→ -t → it means what is type of that content
→ -s → size of ~~commit~~ commit will be character in bytes

13) TOP MOST IS ALWAYS HEAD

↳ So everytime a new commit entered it will become a HEAD.
Every commit has reference of it's past. (last commit).

14) To see difference b/w two specific commit the command is!-

→ git diff <hash code> <hash code>
commit one commit two

you don't need to write complete hash code, it will works with first 5-6 chars of hash code.

15) How hash born? Become

It reads code file and hast it (Hash SHA1) and makes it a commit.

16) What is Hash?

Hash is an algorithm that can content any text or file into a string.

Git reads every file and make it a Hash so it becomes a unique id. Hence, until coder does not make any changes in their code the hash will not change. Once a change/commit occur Hash will be changed because it will check all files and a new change will be shown and that would make a new hash code.

17) MAYA - mostly advanced yet acceptable.
. Advanced.

10) How to go back to a past commit (How to go in past

git reset <hash code of that commit>

above command will change the Head. It will add changes of ~~future~~ (future) ~~to~~ update that were once a present will go into ~~to~~ changes.

in simple - ~~to~~ Those changes will be removed from commit.

for example :-

~~HEAD~~

current status

1) → V1 V2 V3 V4 V5

↳ now this HEAD

2) → Now I want to time travel and go back to V2 then I will use following command :-

, git reset <Hash code of V2 commit>

HEAD

V1 ~~V2~~ V2

Now Head is version 2 V2 and V3 V4 V5 will be removed and they will go into changes. These 3 files as commits & objects will be removed from object folder of .git folder.

19) To permenantely delete or discard latest commit the following command will be used:-

→ git reset --hard <hash code of latest head>

Now those changes can not be taken back. --hard means ~~[illegible]~~ permanently deleted.

20) git log --~~[illegible]~~ oneline

It will show history in one line. All commit and their msgs in seperate lines in a clean way.

21) Config file in .git folder

There is some data in this config file.

92) added command from Github :-

git remote add origin ——— "link"

That above mentioned command will add a link to it's remote.

93) ~~git remote —V~~ git remote —V

It shows Github repo and that means accessing to my teacher's from this :-

PULL

if you want to fetch changes, do from here
if you want to PUSH change, do from here.

PULL → Fetch
PUSH → Send

94) To PUSH

git push —u origin main

-u → upstream

It means push to origin upstream and main branch.

Date |
Page |

VDEO-03) Git and Github Master class - Part - 2 — 29/06/2025

1) Git Branching

Every branch is a different path of Github so Dev can work independly that can be merged later with main branch.

=> How to make a branch ?

a) git branch

↳ it will show name of branch

b) glog

↳ it shows current history line log.

c) git branch <new branch name>

↳ creates a new branch.

2) git checkout <new made branch>

↳ user will move to that branch.

Date
Page

⇒ Merge branch to main branch

a) git merge <new branch name>

→ One command to make new branch and enter in to new branch

a) git checkout -b <branch name>

If branch exist it will enter to that branch otherwise, first it will create a new branch and then enter or move to latest branch.

g) How to add and commit ~~one~~ once.

command → git commit -am "message"

a - added
m - message

git lens and git graph are two extensions of vs code

Date |
Page |

3) ~~Squash~~ Squash

Suppose you created your own branch and then added 3-4 commits, if you add it without squashing, your main branch will get polluted with so many commit ~~that~~ which are not even needed if you can resolve the problem with just one commit.

So while ~~adding~~ merging custom branch to main branch if you added multiple commits in custom branch in main branch it will show one commit.

If you are making a feature, instead adding every commit to ~~the~~ main branch, it is considered better to make one commit than show ~~so~~ about that feature.

⇒ How to do it?

a) first go back to main

↳ git checkout main

b) ↳ git merge --Squash <custom branch name>

↳ this command takes whole custom branch code into single code (local file) but it is heading to commit new changes in main branch using regular commands.

Date |
Page |

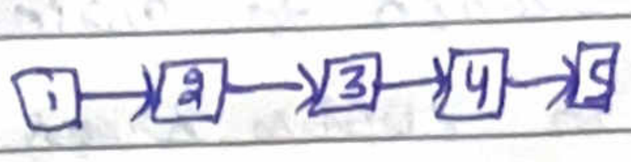
4) Rebase

It re-write the timeline of ~~main~~ branch.

(command -> "(custom branch): git rebase main"

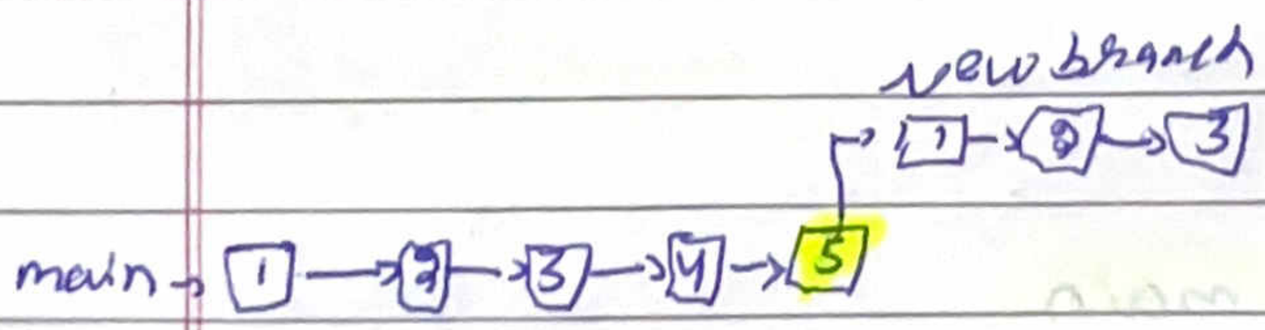
Let's understand with examples:

1) main branch



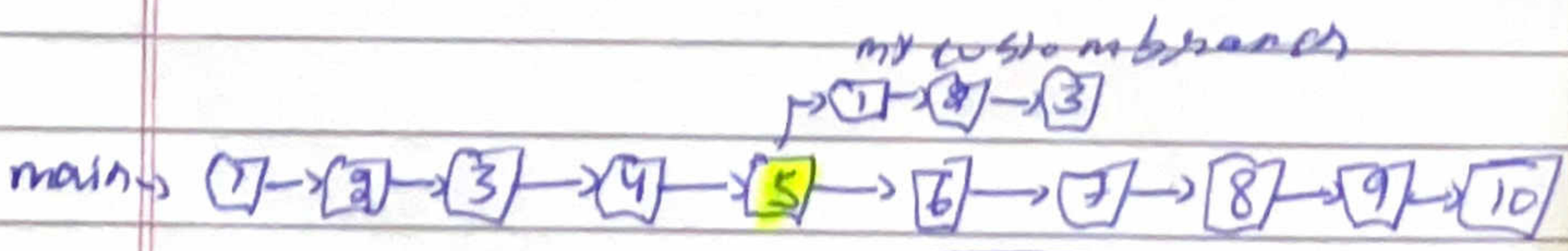
There are Five commits in main branch.

2) creating a new branch



added 3 commits in new branch.

3) Now, I started a new branch from 5th commit of main branch, however, while I was working on some features, other developers add new features in main branch so I left behind in the main branch and I can not my feature in the main branch where I left.



Date |
Page |

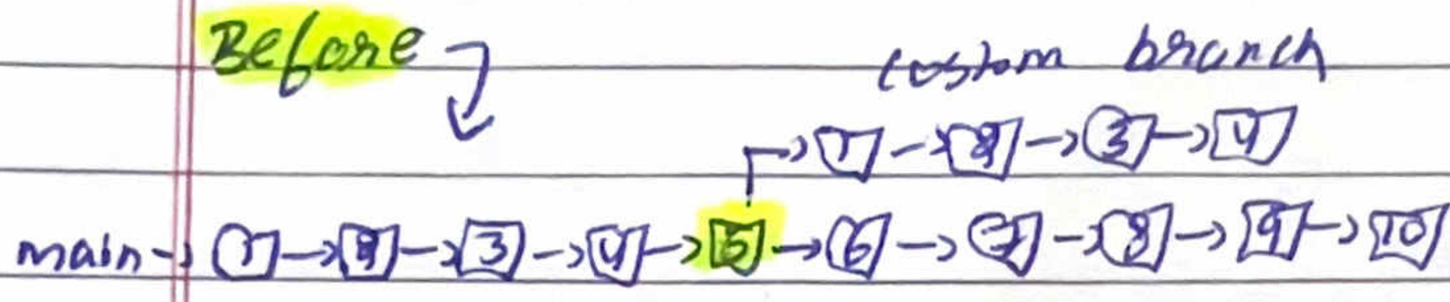
Hence, I can not merge my custom branch because can conflict ch I also can not squash main branch into custom branch ch custom into main branch.

Technically, I can squash branches but if I add custom branch in main branch, it will raise conflicts.

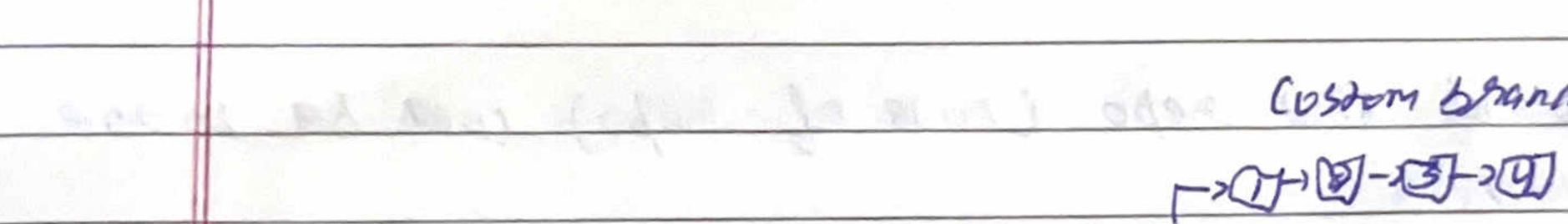
So, easy and one good way to is REBASE.

4) Rebase will rewrite whole timeline.

Before



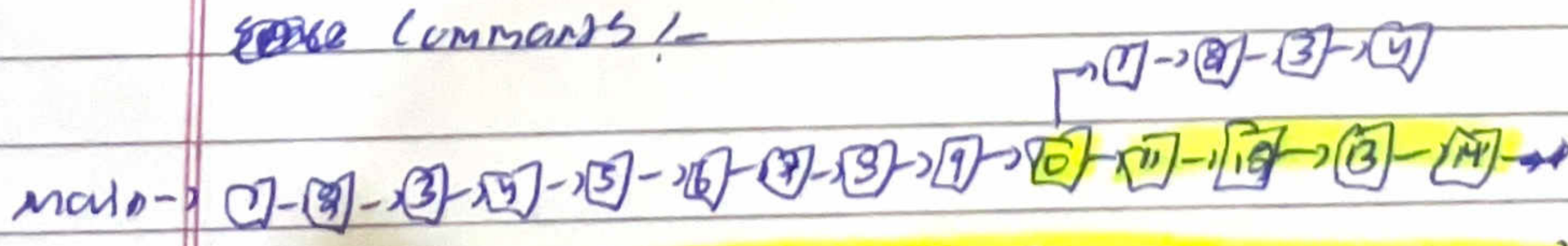
AFTER REBASE



Hence, it will move branch head and its last commit of main branch will become a first commit or head of my custom branch.

Final Reslut

Now, I can merge custom branch into main branch using commands !-



added to main branch. I can squash and add only one commit also in main branch. It depends on company rules.

5) Most Much used Commands

1) git revert
2) git reset
3) git cherry-pick

6) How to make copy of a Repo?

1) Go to Github
2) Find a Repo
3) Click on Fork

7) git clone and url

A copy of that Repo (Code of Repo) will be in the local machine.

PHASE - ☺ is COMPLETED