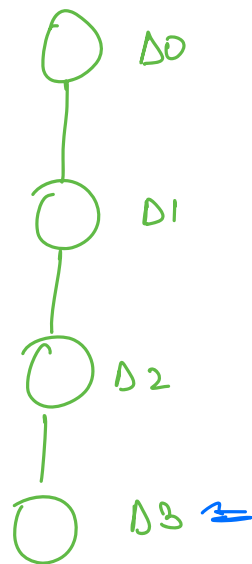Will start at 9:10 PM

Agenda

1. Git Branching
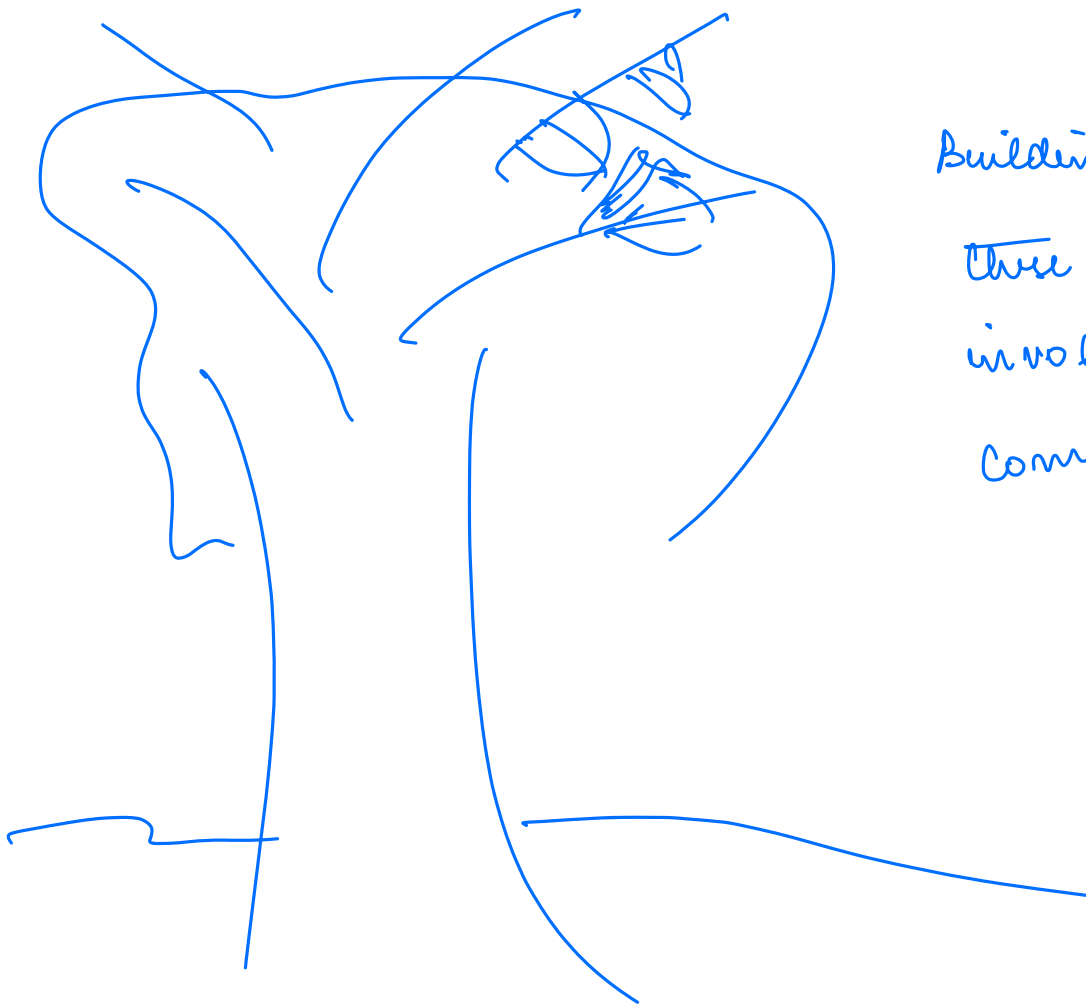2. Merge
3. Rebase
4. Merge v/s Rebase.
5. HEAD in Git

Optional → 6. Git Remote.

else in next
class.

Git Branching
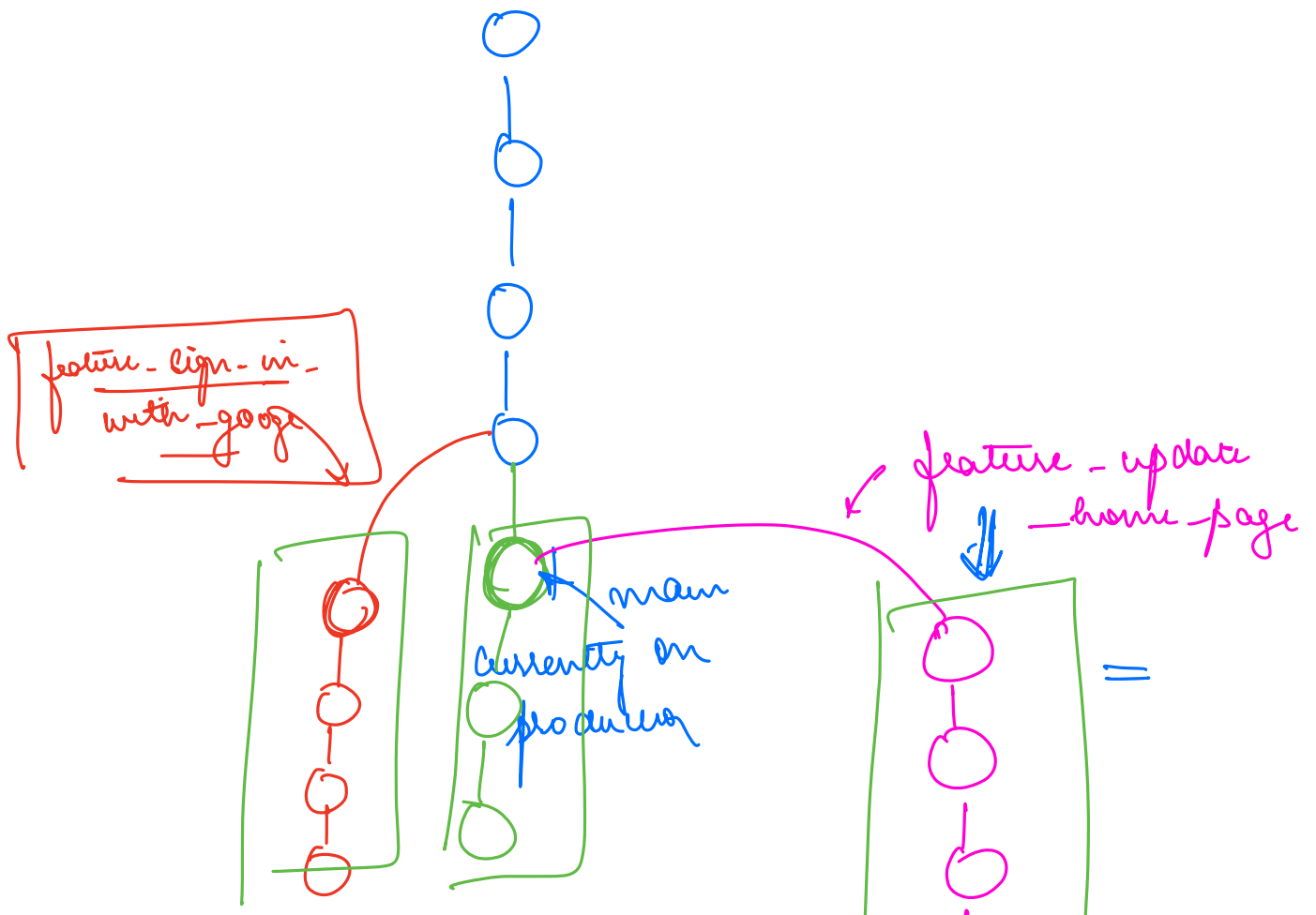

a.) add google key
b.) create of Bul

P1. feature to add
Sign in with
Google on Scaler

P2: Changing the UI
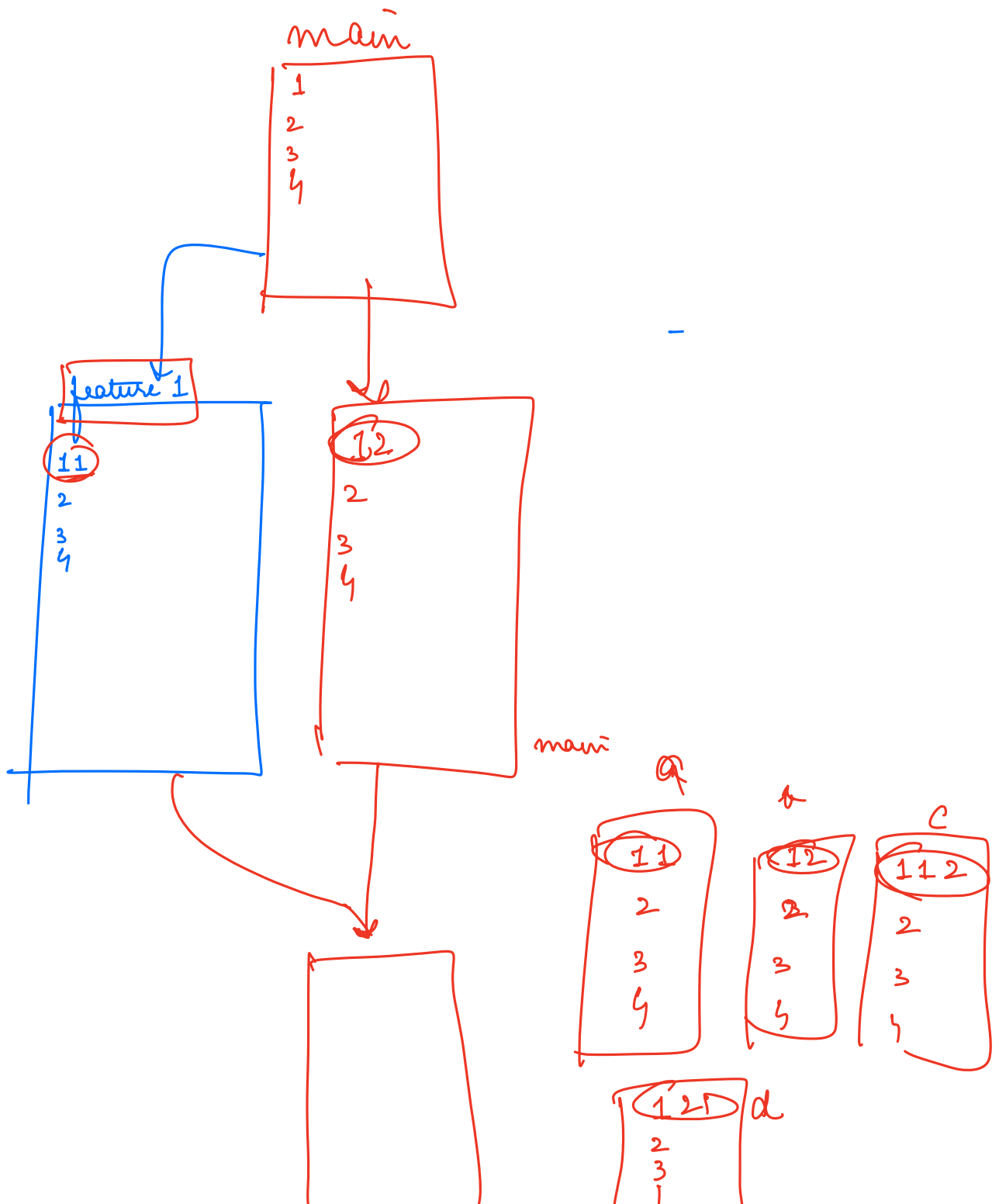of Scaler.com/ academy
a) update instructor
image
b) change colour of

brrll

Building both of these features will involve a lot of commits

feature_sign_in_with_goog

feature_update_home_page

main currently on production

① Create a branch
② Make some change
③ Commit

Branch → parallel line of development

→ when the work of that feature is complete, you would want to merge your changes to production → "main" branch

main

feature 1 ✓

Git Merge

→ command used to bring changes from 1 branch into another branch

main

```
1
2
3
4
```

feature 1

```
1 1
2
3
4
```

```
1 2
2
3
4
```

main

a

```
1 1
2
3
4
```

b

```
1 2
2
3
4
```

c

```
1 1 2
2
3
4
```
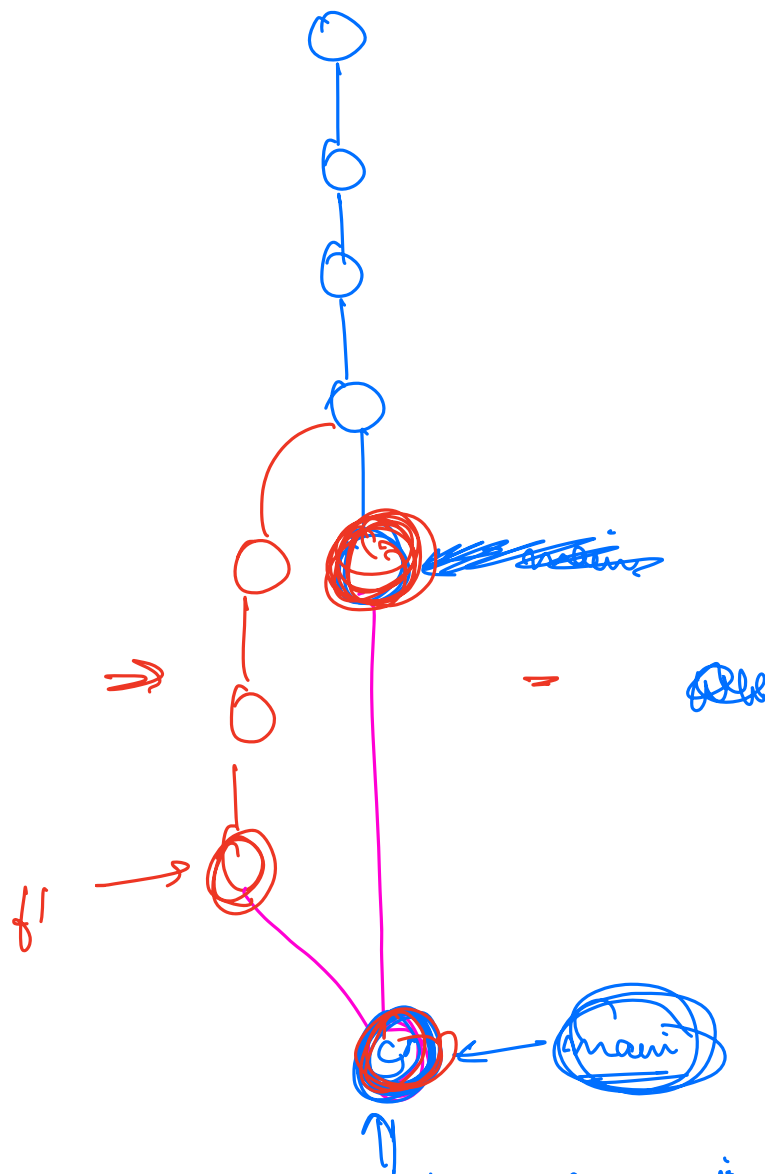
d

```
1 2
2
3
```

2
3
4
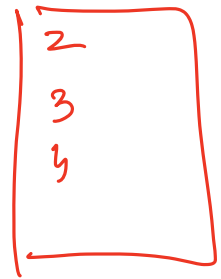
Merge Conflict

When you try to merge all changes of 1 branch into another branch where also some other changes have happened, sometimes there can be a conflict in merging those as git might not be able to judge how final code should look like
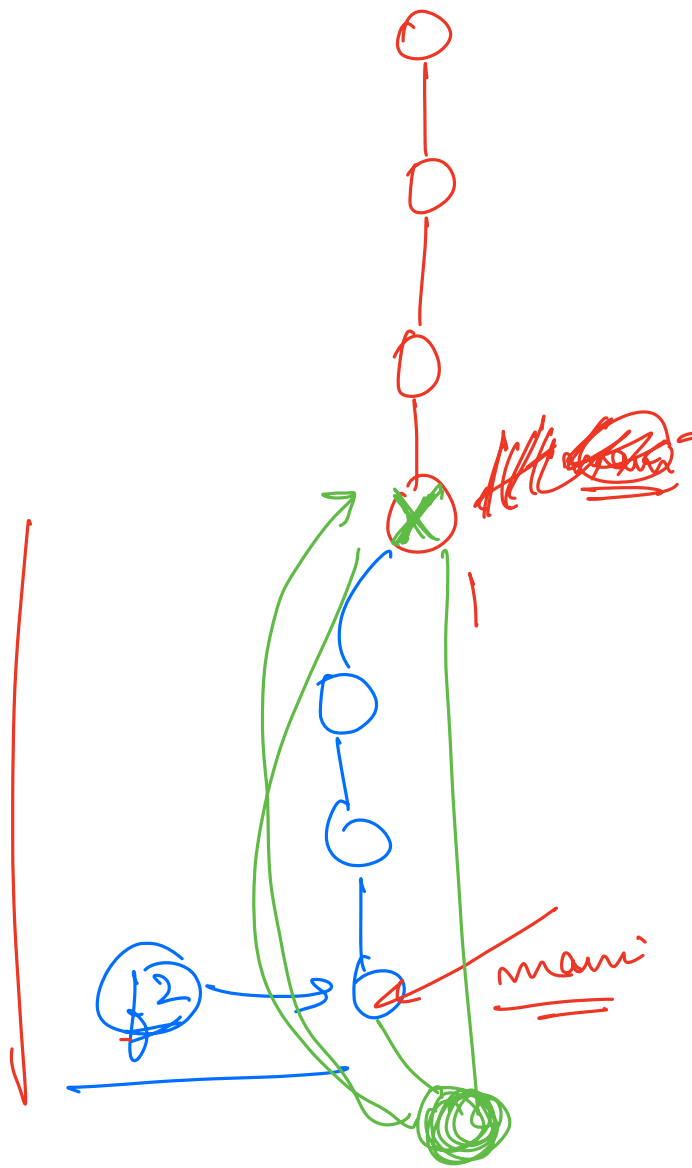
⇒ ambiguity on how final file should look like

main

f1

starting

merge

Merge Commit

Why: Some changes in f1 not in main and vice versa

⇒ A branch is nothing but variable name pointing to a commit

HEAD

main

f2

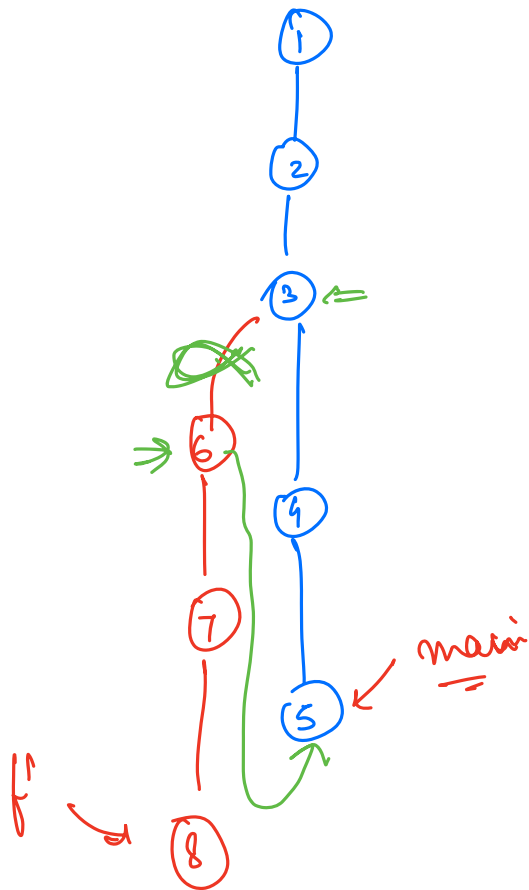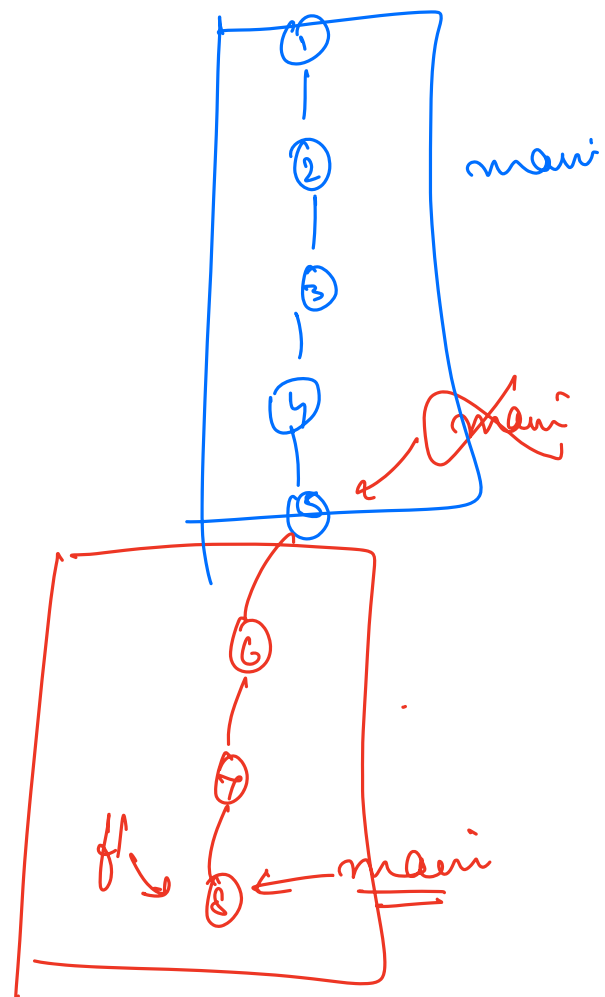⇒ Merge f2 onto main

⇒ I can just move main pointer

⇏ Fast Forward Merge

V/S

main

f'

main

f'

final history in main branch

When you merged 2 branches, production mode

Ideally all changes of f' to be together

Git Rebase allows to remap all the
changes you have done in a
new branch below the latest
change in another branch

① ② ③ ④ ⑤ ⑥ ⑦ ⑧

main

f'

in f'

git rebase main

① ② ③ ④ ⑤ ⑥ ⑦ ⑧

main

main

f'

if you keep doing rebase to main branch often -

when you will merge → fast forward merge

Good Practice ⇒ Rebase → Merge

Direct Merge V/S Rebase → Merge

(a)   (b)

Rebase f1 ONTO main   from f1:
                          git rebase main

Merge   f1 INTO main ⇒ from main;
                          git merge f1

HEAD

→ Special variable in git that points to the _latest commit_

$\Downarrow$
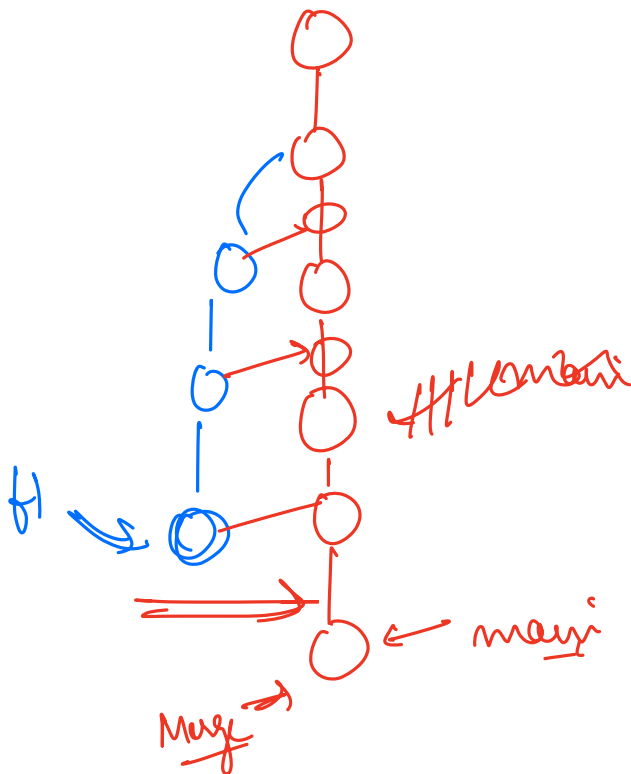
Commit on which any new commit will be made.

Detached HEAD ≠ When HEAD variable isn't pointing to a branch but to a commit

⇒ A new commit may be lost if on a detached head

H/W
1) Read Doc till "HEAD in Git"
2) Do learngit Branching Section 1 completely

Merge: Bring all changes of 1 branch into
another branch

1) Go to main
   git checkout main
2) Bring changes of f1 into main

git merge f1.

f1 ← (blue node)

HEAD main

main

Merge

Rebase

man

is cut out
of main at
X

main

CRDT Conflict free Replicated
       Data Types